

Compilateur

Generated by Doxygen 1.8.11

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	desc_identif Struct Reference	5
3.1.1	Field Documentation	5
3.1.1.1	adresse	5
3.1.1.2	classe	5
3.1.1.3	complement	5
3.1.1.4	identif	5
3.1.1.5	type	5
3.2	dico_ Struct Reference	5
3.2.1	Field Documentation	6
3.2.1.1	base	6
3.2.1.2	sommet	6
3.2.1.3	tab	6
3.3	n_appel_ Struct Reference	6
3.3.1	Field Documentation	6
3.3.1.1	args	6
3.3.1.2	fonction	6
3.4	n_dec_ Struct Reference	6

3.4.1	Member Enumeration Documentation	7
3.4.1.1	anonymous enum	7
3.4.2	Field Documentation	7
3.4.2.1	corps	7
3.4.2.2	foncDec_	7
3.4.2.3	nom	7
3.4.2.4	param	7
3.4.2.5	tabDec_	7
3.4.2.6	taille	7
3.4.2.7	type	7
3.4.2.8	type	8
3.4.2.9	u	8
3.4.2.10	varDec_	8
3.4.2.11	variables	8
3.5	n_exp_ Struct Reference	8
3.5.1	Member Enumeration Documentation	8
3.5.1.1	anonymous enum	8
3.5.2	Field Documentation	9
3.5.2.1	appel	9
3.5.2.2	entier	9
3.5.2.3	incr	9
3.5.2.4	op	9
3.5.2.5	op1	9
3.5.2.6	op2	9
3.5.2.7	opExp_	9
3.5.2.8	type	9
3.5.2.9	u	9
3.5.2.10	var	9
3.6	n_instr_ Struct Reference	9
3.6.1	Member Enumeration Documentation	10

3.6.1.1	anonymous enum	10
3.6.2	Field Documentation	11
3.6.2.1	affecte_	11
3.6.2.2	alors	11
3.6.2.3	appel	11
3.6.2.4	ecrire_	11
3.6.2.5	exp	11
3.6.2.6	expression	11
3.6.2.7	faire	11
3.6.2.8	faire_	11
3.6.2.9	incr	11
3.6.2.10	incr	11
3.6.2.11	init	11
3.6.2.12	liste	11
3.6.2.13	pour_	11
3.6.2.14	retour_	11
3.6.2.15	si_	11
3.6.2.16	sinon	11
3.6.2.17	tantque_	11
3.6.2.18	test	11
3.6.2.19	type	11
3.6.2.20	u	11
3.6.2.21	var	11
3.7	n_l_dec_ Struct Reference	12
3.7.1	Field Documentation	12
3.7.1.1	queue	12
3.7.1.2	tete	12
3.8	n_l_exp_ Struct Reference	12
3.8.1	Field Documentation	12
3.8.1.1	queue	12

3.8.1.2	tete	12
3.9	n_l_instr_ Struct Reference	12
3.9.1	Field Documentation	13
3.9.1.1	queue	13
3.9.1.2	tete	13
3.10	n_prog_ Struct Reference	13
3.10.1	Field Documentation	13
3.10.1.1	fonctions	13
3.10.1.2	variables	13
3.11	n_var_ Struct Reference	13
3.11.1	Member Enumeration Documentation	14
3.11.1.1	anonymous enum	14
3.11.2	Field Documentation	14
3.11.2.1	indice	14
3.11.2.2	indicee_	14
3.11.2.3	nom	14
3.11.2.4	type	14
3.11.2.5	u	14
4	File Documentation	15
4.1	inc/affiche_arbre_abstrait.h File Reference	15
4.1.1	Function Documentation	15
4.1.1.1	affiche_n_prog(n_prog *n)	15
4.2	inc/analyseur_lexical.h File Reference	15
4.2.1	Function Documentation	15
4.2.1.1	nom_token(int token, char *nom, char *valeur)	15
4.2.1.2	test_yylex_internal(FILE *yyin)	15
4.2.1.3	yylex(void)	15
4.3	inc/analyseur_semantique.h File Reference	15
4.3.1	Function Documentation	16
4.3.1.1	semantique(n_prog *p, int trace_dico, int print_mips)	16

4.4	inc/analyseur_syntaxique.h File Reference	16
4.4.1	Function Documentation	17
4.4.1.1	appelFct(void)	17
4.4.1.2	argumentsEffectifs(void)	17
4.4.1.3	comparaison(void)	17
4.4.1.4	comparaisonBis(n_exp *herite)	17
4.4.1.5	conjonction(void)	17
4.4.1.6	conjonctionBis(n_exp *herite)	17
4.4.1.7	declarationFonction(void)	17
4.4.1.8	declarationVariable(void)	17
4.4.1.9	DisplayErreur(void)	17
4.4.1.10	EatTerminal(void)	17
4.4.1.11	expArith(void)	17
4.4.1.12	expArithBis(n_exp *herite)	17
4.4.1.13	expression(void)	17
4.4.1.14	expressionBis(n_exp *herite)	18
4.4.1.15	facteur(void)	18
4.4.1.16	instruction(void)	18
4.4.1.17	instructionAffect(void)	18
4.4.1.18	instructionAppel(void)	18
4.4.1.19	instructionBloc(void)	18
4.4.1.20	instructionEcriture(void)	18
4.4.1.21	instructionFaire(void)	18
4.4.1.22	instructionPour(void)	18
4.4.1.23	instructionRetour(void)	18
4.4.1.24	instructionSi(void)	18
4.4.1.25	instructionTantque(void)	18
4.4.1.26	instructionVide(void)	18
4.4.1.27	listeDecFonctions(void)	18
4.4.1.28	listeDecVariables(void)	18

4.4.1.29	listeDecVariablesBis()	18
4.4.1.30	listeExpressions(void)	18
4.4.1.31	listeExpressionsBis(n_l_exp *herite)	18
4.4.1.32	listeInstructions(void)	18
4.4.1.33	listeParam(void)	18
4.4.1.34	negation(void)	18
4.4.1.35	optDecVariables(void)	18
4.4.1.36	optIndice(void)	18
4.4.1.37	optListeDecVariables(void)	19
4.4.1.38	optSinon(void)	19
4.4.1.39	optTailleTableau(void)	19
4.4.1.40	programme(void)	19
4.4.1.41	syntaxe(int trace_xml_tree)	19
4.4.1.42	terme(void)	19
4.4.1.43	termeBis(n_exp *herite)	19
4.4.1.44	var(void)	19
4.5	inc/dico.h File Reference	19
4.5.1	Macro Definition Documentation	20
4.5.1.1	C_ARGUMENT	20
4.5.1.2	C_VARIABLE_GLOBALE	20
4.5.1.3	C_VARIABLE_LOCALE	20
4.5.1.4	maxDico	20
4.5.1.5	T_ENTIER	20
4.5.1.6	T_FONCTION	20
4.5.1.7	T_TABLEAU_ENTIER	20
4.5.2	Function Documentation	20
4.5.2.1	affiche_dico(void)	20
4.5.2.2	ajouteIdentificateur(char *identif, int classe, int type, int adresse, int complement)	20
4.5.2.3	entreeFonction(void)	20
4.5.2.4	rechercheDeclarative(char *identif)	20

4.5.2.5	rechercheExecutable(char *identif)	20
4.5.2.6	sortieFonction(void)	20
4.5.3	Variable Documentation	20
4.5.3.1	adresseArgumentCourant	20
4.5.3.2	adresseLocaleCourante	20
4.5.3.3	contexte	20
4.5.3.4	dico	20
4.6	inc/freedom.h File Reference	20
4.6.1	Macro Definition Documentation	21
4.6.1.1	__FREEDOM__	21
4.6.2	Function Documentation	21
4.6.2.1	freedom(n_prog *p)	21
4.7	inc/premiers.h File Reference	21
4.7.1	Function Documentation	21
4.7.1.1	est_premier(int non_terminal, int terminal)	21
4.7.1.2	initialise_premiers(void)	21
4.7.2	Variable Documentation	21
4.7.2.1	premiers	21
4.8	inc/suivants.h File Reference	21
4.8.1	Function Documentation	22
4.8.1.1	est_suivant(int non_terminal, int terminal)	22
4.8.1.2	initialise_suivants(void)	22
4.8.2	Variable Documentation	22
4.8.2.1	suivants	22
4.9	inc/symboles.h File Reference	22
4.9.1	Macro Definition Documentation	24
4.9.1.1	_appelFct_	24
4.9.1.2	_argumentsEffectifs_	24
4.9.1.3	_comparaison_	24
4.9.1.4	_comparaisonBis_	24

4.9.1.5	_conjonction_	24
4.9.1.6	_conjonctionBis_	24
4.9.1.7	_declarationFonction_	24
4.9.1.8	_declarationVariable_	24
4.9.1.9	_expArith_	24
4.9.1.10	_expArithBis_	24
4.9.1.11	_expression_	24
4.9.1.12	_expressionBis_	24
4.9.1.13	_facteur_	24
4.9.1.14	_instruction_	24
4.9.1.15	_instructionAffect_	24
4.9.1.16	_instructionAppel_	24
4.9.1.17	_instructionBloc_	24
4.9.1.18	_instructionEcriture_	24
4.9.1.19	_instructionFaire_	24
4.9.1.20	_instructionPour_	24
4.9.1.21	_instructionRetour_	24
4.9.1.22	_instructionSi_	24
4.9.1.23	_instructionTantque_	25
4.9.1.24	_instructionVide_	25
4.9.1.25	_listeDecFonctions_	25
4.9.1.26	_listeDecVariables_	25
4.9.1.27	_listeDecVariablesBis_	25
4.9.1.28	_listeExpressions_	25
4.9.1.29	_listeExpressionsBis_	25
4.9.1.30	_listeInstructions_	25
4.9.1.31	_listeParam_	25
4.9.1.32	_negation_	25
4.9.1.33	_optDecVariables_	25
4.9.1.34	_optIndice_	25

4.9.1.35	<code>_optListeDecVariables_</code>	25
4.9.1.36	<code>_optSinon_</code>	25
4.9.1.37	<code>_optTailleTableau_</code>	25
4.9.1.38	<code>_programme_</code>	25
4.9.1.39	<code>_terme_</code>	25
4.9.1.40	<code>_termeBis_</code>	25
4.9.1.41	<code>_var_</code>	25
4.9.1.42	<code>ACCOLADE_FERMANTE</code>	25
4.9.1.43	<code>ACCOLADE_OUVRANTE</code>	25
4.9.1.44	<code>ALORS</code>	25
4.9.1.45	<code>CROCHET_FERMANT</code>	25
4.9.1.46	<code>CROCHET_OUVRANT</code>	26
4.9.1.47	<code>DIVISE</code>	26
4.9.1.48	<code>ECRIRE</code>	26
4.9.1.49	<code>EGAL</code>	26
4.9.1.50	<code>ENTIER</code>	26
4.9.1.51	<code>EPSILON</code>	26
4.9.1.52	<code>ET</code>	26
4.9.1.53	<code>FAIRE</code>	26
4.9.1.54	<code>FIN</code>	26
4.9.1.55	<code>FOIS</code>	26
4.9.1.56	<code>ID_FCT</code>	26
4.9.1.57	<code>ID_VAR</code>	26
4.9.1.58	<code>INFERIEUR</code>	26
4.9.1.59	<code>LIRE</code>	26
4.9.1.60	<code>MOINS</code>	26
4.9.1.61	<code>NB_NON_TERMINAUX</code>	26
4.9.1.62	<code>NB_TERMINAUX</code>	26
4.9.1.63	<code>NOMBRE</code>	26
4.9.1.64	<code>NON</code>	26

4.9.1.65	OU	26
4.9.1.66	PARENTHESE_FERMANTE	26
4.9.1.67	PARENTHESE_OUVRANTE	26
4.9.1.68	PLUS	26
4.9.1.69	POINT_VIRGULE	27
4.9.1.70	POUR	27
4.9.1.71	RETOUR	27
4.9.1.72	SI	27
4.9.1.73	SINON	27
4.9.1.74	TANTQUE	27
4.9.1.75	VIRGULE	27
4.10	inc/syntax.h File Reference	27
4.10.1	Typedef Documentation	28
4.10.1.1	n_appel	28
4.10.1.2	n_dec	28
4.10.1.3	n_exp	28
4.10.1.4	n_instr	28
4.10.1.5	n_l_dec	28
4.10.1.6	n_l_exp	28
4.10.1.7	n_l_instr	28
4.10.1.8	n_prog	28
4.10.1.9	n_var	28
4.10.2	Enumeration Type Documentation	28
4.10.2.1	operation	28
4.10.3	Function Documentation	29
4.10.3.1	cree_n_appel(char *fonction, n_l_exp *args)	29
4.10.3.2	cree_n_dec_fonc(char *nom, n_l_dec *param, n_l_dec *variables, n_instr *corps)	29
4.10.3.3	cree_n_dec_tab(char *nom, int taille)	29
4.10.3.4	cree_n_dec_var(char *nom)	29
4.10.3.5	cree_n_exp_appel(n_appel *app)	29

4.10.3.6	<code>cree_n_exp_entier(int entier)</code>	29
4.10.3.7	<code>cree_n_exp_incr(n_var *var)</code>	29
4.10.3.8	<code>cree_n_exp_lire(void)</code>	29
4.10.3.9	<code>cree_n_exp_op(operation type, n_exp *op1, n_exp *op2)</code>	29
4.10.3.10	<code>cree_n_exp_var(n_var *var)</code>	29
4.10.3.11	<code>cree_n_instr_affect(n_var *var, n_exp *exp)</code>	29
4.10.3.12	<code>cree_n_instr_appel(n_appel *appel)</code>	29
4.10.3.13	<code>cree_n_instr_bloc(n_l_instr *liste)</code>	29
4.10.3.14	<code>cree_n_instr_ecrire(n_exp *expression)</code>	29
4.10.3.15	<code>cree_n_instr_faire(n_instr *faire, n_exp *test)</code>	30
4.10.3.16	<code>cree_n_instr_incr(n_exp *incr)</code>	30
4.10.3.17	<code>cree_n_instr_pour(n_instr *init, n_exp *test, n_instr *incr, n_instr *faire)</code>	30
4.10.3.18	<code>cree_n_instr_retour(n_exp *expression)</code>	30
4.10.3.19	<code>cree_n_instr_si(n_exp *test, n_instr *alors, n_instr *sinon)</code>	30
4.10.3.20	<code>cree_n_instr_tantque(n_exp *test, n_instr *faire)</code>	30
4.10.3.21	<code>cree_n_instr_vide(void)</code>	30
4.10.3.22	<code>cree_n_l_dec(n_dec *tete, n_l_dec *queue)</code>	30
4.10.3.23	<code>cree_n_l_exp(n_exp *tete, n_l_exp *queue)</code>	30
4.10.3.24	<code>cree_n_l_instr(n_instr *tete, n_l_instr *queue)</code>	30
4.10.3.25	<code>cree_n_prog(n_l_dec *variables, n_l_dec *fonctions)</code>	30
4.10.3.26	<code>cree_n_var_indicee(char *nom, n_exp *indice)</code>	30
4.10.3.27	<code>cree_n_var_simple(char *nom)</code>	30
4.11	<code>inc/util.h</code> File Reference	30
4.11.1	Function Documentation	31
4.11.1.1	<code>affiche_balise_fermante(const char *fct_, int trace_xml)</code>	31
4.11.1.2	<code>affiche_balise_ouvrante(const char *fct_, int trace_xml)</code>	31
4.11.1.3	<code>affiche_element(char *fct_, char *texte_, int trace_xml)</code>	31
4.11.1.4	<code>affiche_texte(char *texte_, int trace_xml)</code>	31
4.11.1.5	<code>duplique_chaine(char *s)</code>	31
4.11.1.6	<code>erreur(char *message)</code>	31

4.11.1.7	<code>erreur_1s(char *message, char *s)</code>	31
4.12	<code>src/affiche_arbre_abstrait.c</code> File Reference	31
4.12.1	Function Documentation	32
4.12.1.1	<code>affiche_appel(n_appel *n)</code>	32
4.12.1.2	<code>affiche_appelExp(n_exp *n)</code>	32
4.12.1.3	<code>affiche_dec(n_dec *n)</code>	32
4.12.1.4	<code>affiche_exp(n_exp *n)</code>	32
4.12.1.5	<code>affiche_foncDec(n_dec *n)</code>	32
4.12.1.6	<code>affiche_instr(n_instr *n)</code>	32
4.12.1.7	<code>affiche_instr_affect(n_instr *n)</code>	32
4.12.1.8	<code>affiche_instr_appel(n_instr *n)</code>	32
4.12.1.9	<code>affiche_instr_ecrire(n_instr *n)</code>	32
4.12.1.10	<code>affiche_instr_faire(n_instr *n)</code>	32
4.12.1.11	<code>affiche_instr_pour(n_instr *n)</code>	32
4.12.1.12	<code>affiche_instr_retour(n_instr *n)</code>	32
4.12.1.13	<code>affiche_instr_si(n_instr *n)</code>	32
4.12.1.14	<code>affiche_instr_tantque(n_instr *n)</code>	32
4.12.1.15	<code>affiche_intExp(n_exp *n)</code>	32
4.12.1.16	<code>affiche_l_dec(n_l_dec *n)</code>	32
4.12.1.17	<code>affiche_l_exp(n_l_exp *n)</code>	32
4.12.1.18	<code>affiche_l_instr(n_l_instr *n)</code>	32
4.12.1.19	<code>affiche_lireExp(n_exp *n)</code>	32
4.12.1.20	<code>affiche_n_prog(n_prog *n)</code>	32
4.12.1.21	<code>affiche_opExp(n_exp *n)</code>	33
4.12.1.22	<code>affiche_tabDec(n_dec *n)</code>	33
4.12.1.23	<code>affiche_var(n_var *n)</code>	33
4.12.1.24	<code>affiche_var_indicee(n_var *n)</code>	33
4.12.1.25	<code>affiche_var_simple(n_var *n)</code>	33
4.12.1.26	<code>affiche_varDec(n_dec *n)</code>	33
4.12.1.27	<code>affiche_varExp(n_exp *n)</code>	33

4.12.2	Variable Documentation	33
4.12.2.1	trace_abs	33
4.13	src/analyseur_lexical.c File Reference	33
4.13.1	Macro Definition Documentation	34
4.13.1.1	is_alpha	34
4.13.1.2	is_alphanum	34
4.13.1.3	is_maj	34
4.13.1.4	is_min	34
4.13.1.5	is_num	34
4.13.1.6	YYTEXT_MAX	34
4.13.2	Function Documentation	34
4.13.2.1	delireCar()	34
4.13.2.2	lireCar(void)	34
4.13.2.3	mangeEspaces()	34
4.13.2.4	nom_token(int token, char *nom, char *valeur)	34
4.13.2.5	test_yylex_internal(FILE *yyin)	34
4.13.2.6	yylex(void)	34
4.13.3	Variable Documentation	34
4.13.3.1	codeMotClefs	34
4.13.3.2	motsClefsMaxLeng	35
4.13.3.3	nb_ligne	35
4.13.3.4	nbMotsClefs	35
4.13.3.5	tableMotsClefs	35
4.13.3.6	yyin	35
4.13.3.7	yylen	35
4.13.3.8	yytext	35
4.14	src/analyseur_semantique.c File Reference	35
4.14.1	Function Documentation	36
4.14.1.1	analyse_appel(n_appel *n)	36
4.14.1.2	analyse_appelExp(n_exp *n)	36

4.14.1.3 analyse_dec(n_dec *n)	36
4.14.1.4 analyse_exp(n_exp *n)	36
4.14.1.5 analyse_foncDec(n_dec *n)	36
4.14.1.6 analyse_instr(n_instr *n)	36
4.14.1.7 analyse_instr_affect(n_instr *n)	37
4.14.1.8 analyse_instr_appel(n_instr *n)	37
4.14.1.9 analyse_instr_ecrire(n_instr *n)	37
4.14.1.10 analyse_instr_faire(n_instr *n)	37
4.14.1.11 analyse_instr_pour(n_instr *n)	37
4.14.1.12 analyse_instr_retour(n_instr *n)	37
4.14.1.13 analyse_instr_si(n_instr *n)	37
4.14.1.14 analyse_instr_tantque(n_instr *n)	37
4.14.1.15 analyse_intExp(n_exp *n)	37
4.14.1.16 analyse_l_dec(n_l_dec *n)	37
4.14.1.17 analyse_l_exp(n_l_exp *n)	37
4.14.1.18 analyse_l_instr(n_l_instr *n)	37
4.14.1.19 analyse_lireExp(n_exp *n)	37
4.14.1.20 analyse_n_prog(n_prog *n)	37
4.14.1.21 analyse_opExp(n_exp *n)	37
4.14.1.22 analyse_tabDec(n_dec *n)	37
4.14.1.23 analyse_var(n_var *n, char *s)	37
4.14.1.24 analyse_var_indicee(n_var *n, char *s)	37
4.14.1.25 analyse_var_simple(n_var *n, char *s)	37
4.14.1.26 analyse_varDec(n_dec *n)	37
4.14.1.27 analyse_varExp(n_exp *n)	37
4.14.1.28 mips_debut_fonction()	37
4.14.1.29 mips_depile(char *s)	37
4.14.1.30 mips_empile(char *s)	37
4.14.1.31 mips_fin_function()	37
4.14.1.32 mips_print(const char *format,...)	37

4.14.1.33	<code>newEtiquette(void)</code>	37
4.14.1.34	<code>semantique(n_prog *p, int trace_dico, int print_mips)</code>	37
4.14.1.35	<code>taille_n_l_dec(n_l_dec *liste)</code>	38
4.14.1.36	<code>taille_n_l_exp(n_l_exp *liste)</code>	38
4.14.2	Variable Documentation	38
4.14.2.1	<code>adresseArgumentCourant</code>	38
4.14.2.2	<code>adresseLocaleCourante</code>	38
4.14.2.3	<code>asreturn</code>	38
4.14.2.4	<code>contexte</code>	38
4.14.2.5	<code>etiquette</code>	38
4.14.2.6	<code>nb_args_function</code>	38
4.14.2.7	<code>nb_var_local</code>	38
4.14.2.8	<code>trace_mips</code>	38
4.14.2.9	<code>trace_tab</code>	38
4.15	<code>src/analyseur_syntaxique.c</code> File Reference	38
4.15.1	Function Documentation	40
4.15.1.1	<code>appelFct(void)</code>	40
4.15.1.2	<code>comparaison(void)</code>	40
4.15.1.3	<code>comparaisonBis(n_exp *herite)</code>	40
4.15.1.4	<code>conjonction(void)</code>	40
4.15.1.5	<code>conjonctionBis(n_exp *herite)</code>	40
4.15.1.6	<code>declarationFonction(void)</code>	40
4.15.1.7	<code>declarationVariable(void)</code>	40
4.15.1.8	<code>DisplayErreur(void)</code>	40
4.15.1.9	<code>EatTerminal(void)</code>	40
4.15.1.10	<code>expArith(void)</code>	40
4.15.1.11	<code>expArithBis(n_exp *herite)</code>	40
4.15.1.12	<code>expression(void)</code>	40
4.15.1.13	<code>expressionBis(n_exp *herite)</code>	40
4.15.1.14	<code>facteur(void)</code>	40

4.15.1.15 instruction(void)	40
4.15.1.16 instructionAffect(void)	40
4.15.1.17 instructionAppel(void)	40
4.15.1.18 instructionBloc(void)	40
4.15.1.19 instructionEcriture(void)	40
4.15.1.20 instructionPour(void)	40
4.15.1.21 instructionRetour(void)	40
4.15.1.22 instructionSi(void)	40
4.15.1.23 instructionTantque(void)	41
4.15.1.24 instructionVide(void)	41
4.15.1.25 listeDecFonctions(void)	41
4.15.1.26 listeDecVariables(void)	41
4.15.1.27 listeDecVariablesBis()	41
4.15.1.28 listeExpressions(void)	41
4.15.1.29 listeExpressionsBis(n_l_exp *herite)	41
4.15.1.30 listeInstructions(void)	41
4.15.1.31 listeParam(void)	41
4.15.1.32 negation(void)	41
4.15.1.33 optDecVariables(void)	41
4.15.1.34 optIndice(void)	41
4.15.1.35 optListeDecVariables(void)	41
4.15.1.36 optSinon(void)	41
4.15.1.37 optTailleTableau(void)	41
4.15.1.38 programme(void)	41
4.15.1.39 syntaxe(int trace_xml_tree)	41
4.15.1.40 terme(void)	41
4.15.1.41 termeBis(n_exp *herite)	41
4.15.1.42 var(void)	41
4.15.2 Variable Documentation	41
4.15.2.1 nb_ligne	41

4.15.2.2	nom	41
4.15.2.3	trace_xml	42
4.15.2.4	uniteCourante	42
4.15.2.5	valeur	42
4.15.2.6	yytext	42
4.16	src/compilateur.c File Reference	42
4.16.1	Function Documentation	42
4.16.1.1	main(int argc, char **argv)	42
4.16.2	Variable Documentation	42
4.16.2.1	yyin	42
4.16.2.2	yytext	42
4.17	src/dico.c File Reference	42
4.17.1	Function Documentation	43
4.17.1.1	affiche_dico(void)	43
4.17.1.2	ajouteIdentificateur(char *identif, int classe, int type, int adresse, int complement)	43
4.17.1.3	entreeFonction(void)	43
4.17.1.4	rechercheDeclarative(char *identif)	43
4.17.1.5	rechercheExecutable(char *identif)	43
4.17.1.6	sortieFonction(void)	43
4.18	src/freedom.c File Reference	43
4.18.1	Function Documentation	44
4.18.1.1	free_appel(n_appel *n)	44
4.18.1.2	free_appelExp(n_exp *n)	44
4.18.1.3	free_dec(n_dec *n)	44
4.18.1.4	free_dico()	44
4.18.1.5	free_exp(n_exp *n)	44
4.18.1.6	free_foncDec(n_dec *n)	44
4.18.1.7	free_instr(n_instr *n)	44
4.18.1.8	free_instr_affect(n_instr *n)	44
4.18.1.9	free_instr_appel(n_instr *n)	44

4.18.1.10 free_instr_ecrire(n_instr *n)	44
4.18.1.11 free_instr_faire(n_instr *n)	44
4.18.1.12 free_instr_pour(n_instr *n)	44
4.18.1.13 free_instr_retour(n_instr *n)	44
4.18.1.14 free_instr_si(n_instr *n)	44
4.18.1.15 free_instr_tantque(n_instr *n)	44
4.18.1.16 free_intExp(n_exp *n)	44
4.18.1.17 free_l_dec(n_l_dec *n)	45
4.18.1.18 free_l_exp(n_l_exp *n)	45
4.18.1.19 free_l_instr(n_l_instr *n)	45
4.18.1.20 free_lireExp(n_exp *n)	45
4.18.1.21 free_n_prog(n_prog *n)	45
4.18.1.22 free_opExp(n_exp *n)	45
4.18.1.23 free_tabDec(n_dec *n)	45
4.18.1.24 free_var(n_var *n)	45
4.18.1.25 free_var_indicee(n_var *n)	45
4.18.1.26 free_var_simple(n_var *n)	45
4.18.1.27 free_varDec(n_dec *n)	45
4.18.1.28 free_varExp(n_exp *n)	45
4.18.1.29 freedom(n_prog *n)	45
4.19 src/premiers.c File Reference	45
4.19.1 Function Documentation	45
4.19.1.1 est_premier(int terminal, int non_terminal)	45
4.19.1.2 initialise_premiers(void)	45
4.20 src/suivants.c File Reference	45
4.20.1 Function Documentation	46
4.20.1.1 est_suivant(int terminal, int non_terminal)	46
4.20.1.2 initialise_suivants(void)	46
4.21 src/syntax.c File Reference	46
4.21.1 Function Documentation	47

4.21.1.1	cree_n_appel(char *fonction, n_l_exp *args)	47
4.21.1.2	cree_n_dec_fonc(char *nom, n_l_dec *param, n_l_dec *variables, n_instr *corps)	47
4.21.1.3	cree_n_dec_tab(char *nom, int taille)	47
4.21.1.4	cree_n_dec_var(char *nom)	47
4.21.1.5	cree_n_exp_appel(n_appel *app)	47
4.21.1.6	cree_n_exp_entier(int entier)	47
4.21.1.7	cree_n_exp_incr(n_var *var)	47
4.21.1.8	cree_n_exp_lire()	47
4.21.1.9	cree_n_exp_op(operation op, n_exp *op1, n_exp *op2)	47
4.21.1.10	cree_n_exp_var(n_var *var)	47
4.21.1.11	cree_n_instr_affect(n_var *var, n_exp *exp)	47
4.21.1.12	cree_n_instr_appel(n_appel *app)	47
4.21.1.13	cree_n_instr_bloc(n_l_instr *liste)	47
4.21.1.14	cree_n_instr_ecrire(n_exp *expression)	47
4.21.1.15	cree_n_instr_faire(n_instr *faire, n_exp *test)	47
4.21.1.16	cree_n_instr_incr(n_exp *incr)	47
4.21.1.17	cree_n_instr_pour(n_instr *init, n_exp *test, n_instr *incr, n_instr *faire)	47
4.21.1.18	cree_n_instr_retour(n_exp *expression)	47
4.21.1.19	cree_n_instr_si(n_exp *test, n_instr *alors, n_instr *sinon)	47
4.21.1.20	cree_n_instr_tantque(n_exp *test, n_instr *faire)	47
4.21.1.21	cree_n_instr_vide(void)	47
4.21.1.22	cree_n_l_dec(n_dec *tete, n_l_dec *queue)	47
4.21.1.23	cree_n_l_exp(n_exp *tete, n_l_exp *queue)	48
4.21.1.24	cree_n_l_instr(n_instr *tete, n_l_instr *queue)	48
4.21.1.25	cree_n_prog(n_l_dec *variables, n_l_dec *fonctions)	48
4.21.1.26	cree_n_var_indicee(char *nom, n_exp *indice)	48
4.21.1.27	cree_n_var_simple(char *nom)	48
4.22	src/util.c File Reference	48
4.22.1	Function Documentation	48
4.22.1.1	affiche_balise_fermante(const char *fct_, int trace_xml)	48
4.22.1.2	affiche_balise_ouvrante(const char *fct_, int trace_xml)	48
4.22.1.3	affiche_element(char *fct_, char *texte_, int trace_xml)	48
4.22.1.4	affiche_texte(char *texte_, int trace_xml)	48
4.22.1.5	affiche_xml_texte(char *texte_)	48
4.22.1.6	duplique_chaine(char *src)	49
4.22.1.7	erreur(char *message)	49
4.22.1.8	erreur_1s(char *message, char *s)	49
4.22.1.9	indent()	49
4.22.2	Variable Documentation	49
4.22.2.1	indent_step	49
4.22.2.2	indent_xml	49
4.22.2.3	nb_ligne	49

Index	51
-----------------------	----

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

desc_identif	5
dico_	5
n_appel_	6
n_dec_	6
n_exp_	8
n_instr_	9
n_l_dec_	12
n_l_exp_	12
n_l_instr_	12
n_prog_	13
n_var_	13

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

inc/affiche_arbre_abstrait.h	15
inc/analyseur_lexical.h	15
inc/analyseur_semantique.h	15
inc/analyseur_syntaxique.h	16
inc/dico.h	19
inc/freedom.h	20
inc/premiers.h	21
inc/suivants.h	21
inc/symboles.h	22
inc/syntax.h	27
inc/util.h	30
src/affiche_arbre_abstrait.c	31
src/analyseur_lexical.c	33
src/analyseur_semantique.c	35
src/analyseur_syntaxique.c	38
src/compilateur.c	42
src/dico.c	42
src/freedom.c	43
src/premiers.c	45
src/suivants.c	45
src/syntax.c	46
src/util.c	48

Chapter 3

Data Structure Documentation

3.1 desc_identif Struct Reference

```
#include <dico.h>
```

Data Fields

- char * [identif](#)
- int [classe](#)
- int [type](#)
- int [adresse](#)
- int [complement](#)

3.1.1 Field Documentation

3.1.1.1 int desc_identif::adresse

3.1.1.2 int desc_identif::classe

3.1.1.3 int desc_identif::complement

3.1.1.4 char* desc_identif::identif

3.1.1.5 int desc_identif::type

The documentation for this struct was generated from the following file:

- inc/[dico.h](#)

3.2 dico_ Struct Reference

```
#include <dico.h>
```

Data Fields

- [desc_identif](#) [tab](#) [[maxDico](#)]
- int [base](#)
- int [sommet](#)

3.2.1 Field Documentation

3.2.1.1 int [dico_::base](#)

3.2.1.2 int [dico_::sommet](#)

3.2.1.3 [desc_identif](#) [dico_::tab](#)[[maxDico](#)]

The documentation for this struct was generated from the following file:

- inc/[dico.h](#)

3.3 n_appel_ Struct Reference

```
#include <syntabs.h>
```

Data Fields

- char * [fonction](#)
- [n_l_exp](#) * [args](#)

3.3.1 Field Documentation

3.3.1.1 [n_l_exp](#)* [n_appel_::args](#)

3.3.1.2 char* [n_appel_::fonction](#)

The documentation for this struct was generated from the following file:

- inc/[syntabs.h](#)

3.4 n_dec_ Struct Reference

```
#include <syntabs.h>
```

Public Types

- enum { [foncDec](#), [varDec](#), [tabDec](#) }

Data Fields

- enum n_dec_:: { ... } [type](#)
- char * [nom](#)
- union {
 - struct {
 - [n_l_dec](#) * [param](#)
 - [n_l_dec](#) * [variables](#)
 - [n_instr](#) * [corps](#)
 - } [foncDec_](#)
 - struct {
 - int [type](#)
 - } [varDec_](#)
 - struct {
 - int [taille](#)
 - } [tabDec_](#)
- } [u](#)

3.4.1 Member Enumeration Documentation

3.4.1.1 anonymous enum

Enumerator

foncDec

varDec

tabDec

3.4.2 Field Documentation

3.4.2.1 [n_instr](#)* [n_dec_::corps](#)

3.4.2.2 struct { ... } [n_dec_::foncDec_](#)

3.4.2.3 char* [n_dec_::nom](#)

3.4.2.4 [n_l_dec](#)* [n_dec_::param](#)

3.4.2.5 struct { ... } [n_dec_::tabDec_](#)

3.4.2.6 int [n_dec_::taille](#)

3.4.2.7 enum { ... } [n_dec_::type](#)

3.4.2.8 `int n_dec_::type`

3.4.2.9 `union { ... } n_dec_::u`

3.4.2.10 `struct { ... } n_dec_::varDec_`

3.4.2.11 `n_l_dec* n_dec_::variables`

The documentation for this struct was generated from the following file:

- [inc/syntaxs.h](#)

3.5 `n_exp_` Struct Reference

```
#include <syntaxs.h>
```

Public Types

- enum {
 [varExp](#), [opExp](#), [intExp](#), [appelExp](#),
 [lireExp](#), [incrExp](#) }

Data Fields

- enum `n_exp_:: { ... } type`
- union {
 struct {
 [operation](#) op
 struct [n_exp_](#) * [op1](#)
 struct [n_exp_](#) * [op2](#)
 } [opExp_](#)
 [n_var](#) * [var](#)
 [n_var](#) * [incr](#)
 int [entier](#)
 [n_appel](#) * [appel](#)
} [u](#)

3.5.1 Member Enumeration Documentation

3.5.1.1 anonymous enum

Enumerator

varExp
opExp
intExp
appelExp
lireExp
incrExp

3.5.2 Field Documentation

3.5.2.1 `n_appel*` `n_exp::appel`

3.5.2.2 `int` `n_exp::entier`

3.5.2.3 `n_var*` `n_exp::incr`

3.5.2.4 `operation` `n_exp::op`

3.5.2.5 `struct n_exp_*` `n_exp::op1`

3.5.2.6 `struct n_exp_*` `n_exp::op2`

3.5.2.7 `struct { ... }` `n_exp::opExp_`

3.5.2.8 `enum { ... }` `n_exp::type`

3.5.2.9 `union { ... }` `n_exp::u`

3.5.2.10 `n_var*` `n_exp::var`

The documentation for this struct was generated from the following file:

- [inc/syntaxs.h](#)

3.6 n_instr_ Struct Reference

```
#include <syntaxs.h>
```

Public Types

- `enum {`
 [incrInst](#), [affecteInst](#), [siInst](#), [faireInst](#),
 [tantqueInst](#), [appellInst](#), [retourInst](#), [ecrireInst](#),
 [videlInst](#), [blocInst](#), [poursInst](#) `}`

Data Fields

- enum n_instr_:: { ... } [type](#)
- union {
 - [n_exp](#) * [incr](#)
 - struct {
 - [n_var](#) * [var](#)
 - [n_exp](#) * [exp](#)
 - } [affecte_](#)
 - struct {
 - [n_exp](#) * [test](#)
 - struct [n_instr_](#) * [alors](#)
 - struct [n_instr_](#) * [sinon](#)
 - } [si_](#)
 - struct {
 - [n_exp](#) * [test](#)
 - struct [n_instr_](#) * [faire](#)
 - } [tantque_](#)
 - struct {
 - [n_exp](#) * [test](#)
 - struct [n_instr_](#) * [faire](#)
 - } [faire_](#)
 - struct {
 - [n_exp](#) * [test](#)
 - struct [n_instr_](#) * [init](#)
 - struct [n_instr_](#) * [incr](#)
 - struct [n_instr_](#) * [faire](#)
 - } [pour_](#)
 - [n_appel](#) * [appel](#)
 - struct {
 - [n_exp](#) * [expression](#)
 - } [retour_](#)
 - struct {
 - [n_exp](#) * [expression](#)
 - } [ecrire_](#)
 - [n_l_instr](#) * [liste](#)
- } [u](#)

3.6.1 Member Enumeration Documentation

3.6.1.1 anonymous enum

Enumerator

incrInst
affecteInst
siInst
faireInst
tantqueInst
appelInst
retourInst
ecrireInst
videInst
blocInst
pourInst

3.6.2 Field Documentation

3.6.2.1 struct { ... } n_instr::affecte_

3.6.2.2 struct n_instr_* n_instr::alors

3.6.2.3 n_appel* n_instr::appel

3.6.2.4 struct { ... } n_instr::ecrire_

3.6.2.5 n_exp* n_instr::exp

3.6.2.6 n_exp* n_instr::expression

3.6.2.7 struct n_instr_* n_instr::faire

3.6.2.8 struct { ... } n_instr::faire_

3.6.2.9 n_exp* n_instr::incr

3.6.2.10 struct n_instr_* n_instr::incr

3.6.2.11 struct n_instr_* n_instr::init

3.6.2.12 n_l_instr* n_instr::liste

3.6.2.13 struct { ... } n_instr::pour_

3.6.2.14 struct { ... } n_instr::retour_

3.6.2.15 struct { ... } n_instr::si_

3.6.2.16 struct n_instr_* n_instr::sinon

3.6.2.17 struct { ... } n_instr::tantque_

3.6.2.18 n_exp* n_instr::test

3.6.2.19 enum { ... } n_instr::type

3.6.2.20 union { ... } n_instr::u

3.6.2.21 n_var* n_instr::var

The documentation for this struct was generated from the following file:

- [inc/syntaxs.h](#)

3.7 n_l_dec_ Struct Reference

```
#include <syntabs.h>
```

Data Fields

- [n_dec](#) * [tete](#)
- struct [n_l_dec_](#) * [queue](#)

3.7.1 Field Documentation

3.7.1.1 struct [n_l_dec_](#) * [n_l_dec_::queue](#)

3.7.1.2 [n_dec](#)* [n_l_dec_::tete](#)

The documentation for this struct was generated from the following file:

- [inc/syntabs.h](#)

3.8 n_l_exp_ Struct Reference

```
#include <syntabs.h>
```

Data Fields

- [n_exp](#) * [tete](#)
- struct [n_l_exp_](#) * [queue](#)

3.8.1 Field Documentation

3.8.1.1 struct [n_l_exp_](#) * [n_l_exp_::queue](#)

3.8.1.2 [n_exp](#)* [n_l_exp_::tete](#)

The documentation for this struct was generated from the following file:

- [inc/syntabs.h](#)

3.9 n_l_instr_ Struct Reference

```
#include <syntabs.h>
```

Data Fields

- [n_instr](#) * [tete](#)
- struct [n_l_instr_](#) * [queue](#)

3.9.1 Field Documentation

3.9.1.1 [struct n_l_instr_](#) * [n_l_instr_::queue](#)

3.9.1.2 [n_instr](#)* [n_l_instr_::tete](#)

The documentation for this struct was generated from the following file:

- [inc/syntaxs.h](#)

3.10 n_prog_ Struct Reference

```
#include <syntaxs.h>
```

Data Fields

- [n_l_dec](#) * [variables](#)
- [n_l_dec](#) * [fonctions](#)

3.10.1 Field Documentation

3.10.1.1 [n_l_dec](#)* [n_prog_::fonctions](#)

3.10.1.2 [n_l_dec](#)* [n_prog_::variables](#)

The documentation for this struct was generated from the following file:

- [inc/syntaxs.h](#)

3.11 n_var_ Struct Reference

```
#include <syntaxs.h>
```

Public Types

- enum { [simple](#), [indicee](#) }

Data Fields

- enum n_var_:: { ... } [type](#)
- char * [nom](#)
- union {
 - struct {
 - [n_exp](#) * [indice](#)
 - [indicee_](#)
- } [u](#)

3.11.1 Member Enumeration Documentation

3.11.1.1 anonymous enum

Enumerator

simple

indicee

3.11.2 Field Documentation

3.11.2.1 [n_exp](#)* [n_var_::indice](#)

3.11.2.2 struct { ... } [n_var_::indicee_](#)

3.11.2.3 char* [n_var_::nom](#)

3.11.2.4 enum { ... } [n_var_::type](#)

3.11.2.5 union { ... } [n_var_::u](#)

The documentation for this struct was generated from the following file:

- [inc/syntaxs.h](#)

Chapter 4

File Documentation

4.1 inc/affiche_arbre_abstrait.h File Reference

```
#include "syntabs.h"
```

Functions

- void [affiche_n_prog](#) ([n_prog](#) *n)

4.1.1 Function Documentation

4.1.1.1 void [affiche_n_prog](#) ([n_prog](#) * *n*)

4.2 inc/analyseur_lexical.h File Reference

```
#include "stdio.h"
```

Functions

- int [yylex](#) (void)
- void [nom_token](#) (int token, char *[nom](#), char *[valeur](#))
- void [test_yylex_internal](#) (FILE *[yyin](#))

4.2.1 Function Documentation

4.2.1.1 void [nom_token](#) (int *token*, char * *nom*, char * *valeur*)

4.2.1.2 void [test_yylex_internal](#) (FILE * *yyin*)

4.2.1.3 int [yylex](#) (void)

4.3 inc/analyseur_semantique.h File Reference

```
#include "syntabs.h"
```

Functions

- void [semantique](#) ([n_prog](#) *p, int trace_dico, int print_mips)
Main function to parse syntaxique tree.

4.3.1 Function Documentation

4.3.1.1 void [semantique](#) ([n_prog](#) * p, int *trace_dico*, int *print_mips*)

Main function to parse syntaxique tree.

Parameters

	<i>p</i>	The head of the tree
in	<i>trace_dico</i>	boolean to display symbol table
in	<i>print_mips</i>	boolean to display the mips output on stdin

4.4 inc/analyseur_syntaxique.h File Reference

```
#include "syntabs.h"
```

Functions

- [n_l_dec](#) * [listeDecVariables](#) (void)
- [n_l_dec](#) * [listeDecFonctions](#) (void)
- [n_dec](#) * [declarationVariable](#) (void)
- [n_dec](#) * [declarationFonction](#) (void)
- [n_l_dec](#) * [listeParam](#) (void)
- [n_l_instr](#) * [listeInstructions](#) (void)
- [n_instr](#) * [instruction](#) (void)
- [n_instr](#) * [instructionAffect](#) (void)
- [n_instr](#) * [instructionBloc](#) (void)
- [n_instr](#) * [instructionSi](#) (void)
- [n_instr](#) * [instructionTantque](#) (void)
- [n_instr](#) * [instructionAppel](#) (void)
- [n_instr](#) * [instructionRetour](#) (void)
- [n_instr](#) * [instructionEcriture](#) (void)
- [n_instr](#) * [instructionVide](#) (void)
- [n_var](#) * [var](#) (void)
- [n_exp](#) * [expression](#) (void)
- [n_appel](#) * [appelFct](#) (void)
- [n_exp](#) * [conjonction](#) (void)
- [n_exp](#) * [negation](#) (void)
- [n_exp](#) * [comparaison](#) (void)
- [n_exp](#) * [expArith](#) (void)
- [n_exp](#) * [terme](#) (void)
- [n_exp](#) * [facteur](#) (void)

- [n_exp * argumentsEffectifs](#) (void)
- [n_l_exp * listeExpressions](#) (void)
- [n_l_exp * listeExpressionsBis](#) ([n_l_exp](#) *herite)
- [n_prog * programme](#) (void)
- [n_exp * conjonctionBis](#) ([n_exp](#) *herite)
- [int optTailleTableau](#) (void)
- [n_exp * expArithBis](#) ([n_exp](#) *herite)
- [n_instr * optSinon](#) (void)
- [n_exp * comparaisonBis](#) ([n_exp](#) *herite)
- [n_l_dec * optDecVariables](#) (void)
- [n_exp * optIndice](#) (void)
- [n_l_dec * listeDecVariablesBis](#) ()
- [n_exp * termeBis](#) ([n_exp](#) *herite)
- [n_exp * expressionBis](#) ([n_exp](#) *herite)
- [n_instr * instructionFaire](#) (void)
- [n_l_dec * optListeDecVariables](#) (void)
- [n_instr * instructionPour](#) (void)
- [void EatTerminal](#) (void)
- [void DisplayErreur](#) (void)
- [n_prog * syntaxe](#) (int trace_xml_tree)

4.4.1 Function Documentation

4.4.1.1 [n_appel* appelFct](#) (void)

4.4.1.2 [n_exp* argumentsEffectifs](#) (void)

4.4.1.3 [n_exp* comparaison](#) (void)

4.4.1.4 [n_exp* comparaisonBis](#) ([n_exp](#) * herite)

4.4.1.5 [n_exp* conjonction](#) (void)

4.4.1.6 [n_exp* conjonctionBis](#) ([n_exp](#) * herite)

4.4.1.7 [n_dec* declarationFonction](#) (void)

4.4.1.8 [n_dec* declarationVariable](#) (void)

4.4.1.9 [void DisplayErreur](#) (void)

4.4.1.10 [void EatTerminal](#) (void)

4.4.1.11 [n_exp* expArith](#) (void)

4.4.1.12 [n_exp* expArithBis](#) ([n_exp](#) * herite)

4.4.1.13 [n_exp* expression](#) (void)

- 4.4.1.14 `n_exp* expressionBis (n_exp * herite)`
- 4.4.1.15 `n_exp* facteur (void)`
- 4.4.1.16 `n_instr* instruction (void)`
- 4.4.1.17 `n_instr* instructionAffect (void)`
- 4.4.1.18 `n_instr* instructionAppel (void)`
- 4.4.1.19 `n_instr* instructionBloc (void)`
- 4.4.1.20 `n_instr* instructionEcriture (void)`
- 4.4.1.21 `n_instr* instructionFaire (void)`
- 4.4.1.22 `n_instr* instructionPour (void)`
- 4.4.1.23 `n_instr* instructionRetour (void)`
- 4.4.1.24 `n_instr* instructionSi (void)`
- 4.4.1.25 `n_instr* instructionTantque (void)`
- 4.4.1.26 `n_instr* instructionVide (void)`
- 4.4.1.27 `n_l_dec* listeDecFonctions (void)`
- 4.4.1.28 `n_l_dec* listeDecVariables (void)`
- 4.4.1.29 `n_l_dec* listeDecVariablesBis ()`
- 4.4.1.30 `n_l_exp* listeExpressions (void)`
- 4.4.1.31 `n_l_exp* listeExpressionsBis (n_l_exp * herite)`
- 4.4.1.32 `n_l_instr* listeInstructions (void)`
- 4.4.1.33 `n_l_dec* listeParam (void)`
- 4.4.1.34 `n_exp* negation (void)`
- 4.4.1.35 `n_l_dec* optDecVariables (void)`
- 4.4.1.36 `n_exp* optIndice (void)`

4.4.1.37 `n_l_dec* optListeDecVariables (void)`

4.4.1.38 `n_instr* optSinon (void)`

4.4.1.39 `int optTailleTableau (void)`

4.4.1.40 `n_prog* programme (void)`

4.4.1.41 `n_prog* syntaxe (int trace_xml_tree)`

4.4.1.42 `n_exp* terme (void)`

4.4.1.43 `n_exp* termeBis (n_exp * herite)`

4.4.1.44 `n_var* var (void)`

4.5 inc/dico.h File Reference

```
#include "syntabs.h"
```

Data Structures

- struct [desc_identif](#)
- struct [dico_](#)

Macros

- #define [maxDico](#) 100
- #define [C_VARIABLE_GLOBALE](#) 1
- #define [C_VARIABLE_LOCALE](#) 2
- #define [C_ARGUMENT](#) 3
- #define [T_ENTIER](#) 1
- #define [T_TABLEAU_ENTIER](#) 2
- #define [T_FONCTION](#) 3

Functions

- void [entreeFonction](#) (void)
- void [sortieFonction](#) (void)
- int [ajouteIdentificateur](#) (char *identif, int classe, int type, int adresse, int complement)
- int [rechercheExecutable](#) (char *identif)
- int [rechercheDeclarative](#) (char *identif)
- void [affiche_dico](#) (void)

Variables

- dico_dico
- int contexte
- int adresseLocaleCourante
- int adresseArgumentCourant

4.5.1 Macro Definition Documentation

4.5.1.1 #define C_ARGUMENT 3

4.5.1.2 #define C_VARIABLE_GLOBALE 1

4.5.1.3 #define C_VARIABLE_LOCALE 2

4.5.1.4 #define maxDico 100

4.5.1.5 #define T_ENTIER 1

4.5.1.6 #define T_FONCTION 3

4.5.1.7 #define T_TABLEAU_ENTIER 2

4.5.2 Function Documentation

4.5.2.1 void affiche_dico (void)

4.5.2.2 int ajoutIdentificateur (char * *identif*, int *classe*, int *type*, int *adresse*, int *complement*)

4.5.2.3 void entreeFonction (void)

4.5.2.4 int rechercheDeclarative (char * *identif*)

4.5.2.5 int rechercheExecutable (char * *identif*)

4.5.2.6 void sortieFonction (void)

4.5.3 Variable Documentation

4.5.3.1 int adresseArgumentCourant

4.5.3.2 int adresseLocaleCourante

4.5.3.3 int contexte

4.5.3.4 dico_dico

4.6 inc/freedom.h File Reference

```
#include "syntabs.h"
```

Macros

- #define `__FREEDOM__` value

Functions

- void `freedom` (`n_prog` *`p`)

4.6.1 Macro Definition Documentation

4.6.1.1 #define `__FREEDOM__` value

4.6.2 Function Documentation

4.6.2.1 void `freedom` (`n_prog` * `p`)

4.7 inc/premiers.h File Reference

```
#include "symboles.h"
```

Functions

- void `initialise_premiers` (void)
- int `est_premier` (int `non_terminal`, int `terminal`)

Variables

- int `premiers` [`NB_NON_TERMINAUX`+1][`NB_TERMINAUX`+1]

4.7.1 Function Documentation

4.7.1.1 int `est_premier` (int `non_terminal`, int `terminal`)

4.7.1.2 void `initialise_premiers` (void)

4.7.2 Variable Documentation

4.7.2.1 int `premiers`[`NB_NON_TERMINAUX`+1][`NB_TERMINAUX`+1]

4.8 inc/suivants.h File Reference

```
#include "symboles.h"
```

Functions

- void [initialise_suivants](#) (void)
- int [est_suivant](#) (int non_terminal, int terminal)

Variables

- int [suivants](#) [[NB_NON_TERMINAUX](#)+1][[NB_TERMINAUX](#)+1]

4.8.1 Function Documentation

4.8.1.1 int [est_suivant](#) (int *non_terminal*, int *terminal*)

4.8.1.2 void [initialise_suivants](#) (void)

4.8.2 Variable Documentation

4.8.2.1 int [suivants](#)[[NB_NON_TERMINAUX](#)+1][[NB_TERMINAUX](#)+1]

4.9 inc/symboles.h File Reference

Macros

- #define [EPSILON](#) 0
- #define [NB_NON_TERMINAUX](#) 43
- #define [_listeDecVariables_](#) 1
- #define [_listeDecFonctions_](#) 2
- #define [_declarationVariable_](#) 3
- #define [_declarationFonction_](#) 4
- #define [_listeParam_](#) 5
- #define [_listeInstructions_](#) 6
- #define [_instruction_](#) 8
- #define [_instructionAffect_](#) 9
- #define [_instructionBloc_](#) 10
- #define [_instructionSi_](#) 11
- #define [_instructionTantque_](#) 12
- #define [_instructionAppel_](#) 13
- #define [_instructionRetour_](#) 14
- #define [_instructionEcriture_](#) 15
- #define [_instructionVide_](#) 16
- #define [_var_](#) 17
- #define [_expression_](#) 18
- #define [_appelFct_](#) 19
- #define [_conjonction_](#) 20
- #define [_negation_](#) 21
- #define [_comparaison_](#) 22
- #define [_expArith_](#) 23
- #define [_terme_](#) 24
- #define [_facteur_](#) 25

- [#define _argumentsEffectifs_ 26](#)
- [#define _listeExpressions_ 27](#)
- [#define _listeExpressionsBis_ 7](#)
- [#define _programme_ 28](#)
- [#define _conjonctionBis_ 29](#)
- [#define _optTailleTableau_ 30](#)
- [#define _expArithBis_ 31](#)
- [#define _optSinon_ 32](#)
- [#define _comparaisonBis_ 33](#)
- [#define _optDecVariables_ 34](#)
- [#define _optIndice_ 35](#)
- [#define _listeDecVariablesBis_ 36](#)
- [#define _termeBis_ 38](#)
- [#define _expressionBis_ 39](#)
- [#define _instructionFaire_ 40](#)
- [#define _optListeDecVariables_ 41](#)
- [#define _instructionPour_ 42](#)
- [#define NB_TERMINAUX 32](#)
- [#define POINT_VIRGULE 1](#)
- [#define PLUS 2](#)
- [#define MOINS 3](#)
- [#define FOIS 4](#)
- [#define DIVISE 5](#)
- [#define PARENTHESE_OUVRANTE 6](#)
- [#define PARENTHESE_FERMANTE 7](#)
- [#define CROCHET_OUVRANT 8](#)
- [#define CROCHET_FERMANT 9](#)
- [#define ACCOLADE_OUVRANTE 10](#)
- [#define ACCOLADE_FERMANTE 11](#)
- [#define EGAL 12](#)
- [#define INFERIEUR 13](#)
- [#define ET 14](#)
- [#define OU 15](#)
- [#define NON 16](#)
- [#define SI 17](#)
- [#define ALORS 18](#)
- [#define SINON 19](#)
- [#define TANTQUE 20](#)
- [#define FAIRE 21](#)
- [#define ENTIER 22](#)
- [#define RETOUR 23](#)
- [#define LIRE 24](#)
- [#define ECRIRE 25](#)
- [#define ID_VAR 26](#)
- [#define ID_FCT 27](#)
- [#define NOMBRE 28](#)
- [#define FIN 29](#)
- [#define VIRGULE 30](#)
- [#define POUR 31](#)

4.9.1 Macro Definition Documentation

4.9.1.1 `#define _appelFct_ 19`

4.9.1.2 `#define _argumentsEffectifs_ 26`

4.9.1.3 `#define _comparaison_ 22`

4.9.1.4 `#define _comparaisonBis_ 33`

4.9.1.5 `#define _conjonction_ 20`

4.9.1.6 `#define _conjonctionBis_ 29`

4.9.1.7 `#define _declarationFonction_ 4`

4.9.1.8 `#define _declarationVariable_ 3`

4.9.1.9 `#define _expArith_ 23`

4.9.1.10 `#define _expArithBis_ 31`

4.9.1.11 `#define _expression_ 18`

4.9.1.12 `#define _expressionBis_ 39`

4.9.1.13 `#define _facteur_ 25`

4.9.1.14 `#define _instruction_ 8`

4.9.1.15 `#define _instructionAffect_ 9`

4.9.1.16 `#define _instructionAppel_ 13`

4.9.1.17 `#define _instructionBloc_ 10`

4.9.1.18 `#define _instructionEcriture_ 15`

4.9.1.19 `#define _instructionFaire_ 40`

4.9.1.20 `#define _instructionPour_ 42`

4.9.1.21 `#define _instructionRetour_ 14`

4.9.1.22 `#define _instructionSi_ 11`

- 4.9.1.23 `#define _instructionTantque_ 12`
- 4.9.1.24 `#define _instructionVide_ 16`
- 4.9.1.25 `#define _listeDecFonctions_ 2`
- 4.9.1.26 `#define _listeDecVariables_ 1`
- 4.9.1.27 `#define _listeDecVariablesBis_ 36`
- 4.9.1.28 `#define _listeExpressions_ 27`
- 4.9.1.29 `#define _listeExpressionsBis_ 7`
- 4.9.1.30 `#define _listeInstructions_ 6`
- 4.9.1.31 `#define _listeParam_ 5`
- 4.9.1.32 `#define _negation_ 21`
- 4.9.1.33 `#define _optDecVariables_ 34`
- 4.9.1.34 `#define _optIndice_ 35`
- 4.9.1.35 `#define _optListeDecVariables_ 41`
- 4.9.1.36 `#define _optSinon_ 32`
- 4.9.1.37 `#define _optTailleTableau_ 30`
- 4.9.1.38 `#define _programme_ 28`
- 4.9.1.39 `#define _terme_ 24`
- 4.9.1.40 `#define _termeBis_ 38`
- 4.9.1.41 `#define _var_ 17`
- 4.9.1.42 `#define ACCOLADE_FERMANTE 11`
- 4.9.1.43 `#define ACCOLADE_OUVRANTE 10`
- 4.9.1.44 `#define ALORS 18`
- 4.9.1.45 `#define CROCHET_FERMANT 9`

4.9.1.46 `#define CROCHET_OUVRANT 8`

4.9.1.47 `#define DIVISE 5`

4.9.1.48 `#define ECRIRE 25`

4.9.1.49 `#define EGAL 12`

4.9.1.50 `#define ENTIER 22`

4.9.1.51 `#define EPSILON 0`

4.9.1.52 `#define ET 14`

4.9.1.53 `#define FAIRE 21`

4.9.1.54 `#define FIN 29`

4.9.1.55 `#define FOIS 4`

4.9.1.56 `#define ID_FCT 27`

4.9.1.57 `#define ID_VAR 26`

4.9.1.58 `#define INFERIEUR 13`

4.9.1.59 `#define LIRE 24`

4.9.1.60 `#define MOINS 3`

4.9.1.61 `#define NB_NON_TERMINAUX 43`

4.9.1.62 `#define NB_TERMINAUX 32`

4.9.1.63 `#define NOMBRE 28`

4.9.1.64 `#define NON 16`

4.9.1.65 `#define OU 15`

4.9.1.66 `#define PARENTHESE_FERMANTE 7`

4.9.1.67 `#define PARENTHESE_OUVRANTE 6`

4.9.1.68 `#define PLUS 2`

4.9.1.69 `#define POINT_VIRGULE 1`

4.9.1.70 `#define POUR 31`

4.9.1.71 `#define RETOUR 23`

4.9.1.72 `#define SI 17`

4.9.1.73 `#define SINON 19`

4.9.1.74 `#define TANTQUE 20`

4.9.1.75 `#define VIRGULE 30`

4.10 inc/syntax.h File Reference

Data Structures

- struct [n_prog_](#)
- struct [n_dec_](#)
- struct [n_exp_](#)
- struct [n_instr_](#)
- struct [n_appel_](#)
- struct [n_var_](#)
- struct [n_l_exp_](#)
- struct [n_l_instr_](#)
- struct [n_l_dec_](#)

Typedefs

- typedef struct [n_l_instr_ n_l_instr](#)
- typedef struct [n_instr_ n_instr](#)
- typedef struct [n_exp_ n_exp](#)
- typedef struct [n_l_exp_ n_l_exp](#)
- typedef struct [n_var_ n_var](#)
- typedef struct [n_l_dec_ n_l_dec](#)
- typedef struct [n_dec_ n_dec](#)
- typedef struct [n_prog_ n_prog](#)
- typedef struct [n_appel_ n_appel](#)

Enumerations

- enum [operation](#) {
 [plus](#), [moins](#), [fois](#), [divise](#),
 [modulo](#), [egal](#), [diff](#), [inf](#),
 [sup](#), [infeg](#), [supeg](#), [ou](#),
 [et](#), [non](#), [negatif](#) }

Functions

- `n_prog` * `cree_n_prog` (`n_l_dec` *variables, `n_l_dec` *fonctions)
- `n_dec` * `cree_n_dec_var` (char *nom)
- `n_dec` * `cree_n_dec_tab` (char *nom, int taille)
- `n_dec` * `cree_n_dec_fonc` (char *nom, `n_l_dec` *param, `n_l_dec` *variables, `n_instr` *corps)
- `n_exp` * `cree_n_exp_op` (operation type, `n_exp` *op1, `n_exp` *op2)
- `n_exp` * `cree_n_exp_entier` (int entier)
- `n_exp` * `cree_n_exp_var` (`n_var` *var)
- `n_exp` * `cree_n_exp_appel` (`n_appel` *app)
- `n_exp` * `cree_n_exp_lire` (void)
- `n_exp` * `cree_n_exp_incr` (`n_var` *var)
- `n_instr` * `cree_n_instr_incr` (`n_exp` *incr)
- `n_instr` * `cree_n_instr_si` (`n_exp` *test, `n_instr` *alors, `n_instr` *sinon)
- `n_instr` * `cree_n_instr_bloc` (`n_l_instr` *liste)
- `n_instr` * `cree_n_instr_tantque` (`n_exp` *test, `n_instr` *faire)
- `n_instr` * `cree_n_instr_faire` (`n_instr` *faire, `n_exp` *test)
- `n_instr` * `cree_n_instr_pour` (`n_instr` *init, `n_exp` *test, `n_instr` *incr, `n_instr` *faire)
- `n_instr` * `cree_n_instr_affect` (`n_var` *var, `n_exp` *exp)
- `n_instr` * `cree_n_instr_appel` (`n_appel` *appel)
- `n_instr` * `cree_n_instr_retour` (`n_exp` *expression)
- `n_instr` * `cree_n_instr_ecrire` (`n_exp` *expression)
- `n_instr` * `cree_n_instr_vide` (void)
- `n_appel` * `cree_n_appel` (char *fonction, `n_l_exp` *args)
- `n_var` * `cree_n_var_simple` (char *nom)
- `n_var` * `cree_n_var_indicee` (char *nom, `n_exp` *indice)
- `n_l_exp` * `cree_n_l_exp` (`n_exp` *tete, `n_l_exp` *queue)
- `n_l_instr` * `cree_n_l_instr` (`n_instr` *tete, `n_l_instr` *queue)
- `n_l_dec` * `cree_n_l_dec` (`n_dec` *tete, `n_l_dec` *queue)

4.10.1 Typedef Documentation

4.10.1.1 typedef struct `n_appel_n_appel`

4.10.1.2 typedef struct `n_dec_n_dec`

4.10.1.3 typedef struct `n_exp_n_exp`

4.10.1.4 typedef struct `n_instr_n_instr`

4.10.1.5 typedef struct `n_l_dec_n_l_dec`

4.10.1.6 typedef struct `n_l_exp_n_l_exp`

4.10.1.7 typedef struct `n_l_instr_n_l_instr`

4.10.1.8 typedef struct `n_prog_n_prog`

4.10.1.9 typedef struct `n_var_n_var`

4.10.2 Enumeration Type Documentation

4.10.2.1 enum operation

Enumerator

plus

moins
fois
divise
modulo
egal
diff
inf
sup
infeg
supeg
ou
et
non
negatif

4.10.3 Function Documentation

4.10.3.1 `n_appel*` `cree_n_appel` (`char *` *fonction*, `n_l_exp *` *args*)

4.10.3.2 `n_dec*` `cree_n_dec_fonc` (`char *` *nom*, `n_l_dec *` *param*, `n_l_dec *` *variables*, `n_instr *` *corps*)

4.10.3.3 `n_dec*` `cree_n_dec_tab` (`char *` *nom*, `int` *taille*)

4.10.3.4 `n_dec*` `cree_n_dec_var` (`char *` *nom*)

4.10.3.5 `n_exp*` `cree_n_exp_appel` (`n_appel *` *app*)

4.10.3.6 `n_exp*` `cree_n_exp_entier` (`int` *entier*)

4.10.3.7 `n_exp*` `cree_n_exp_incr` (`n_var *` *var*)

4.10.3.8 `n_exp*` `cree_n_exp_lire` (`void`)

4.10.3.9 `n_exp*` `cree_n_exp_op` (*operation type*, `n_exp *` *op1*, `n_exp *` *op2*)

4.10.3.10 `n_exp*` `cree_n_exp_var` (`n_var *` *var*)

4.10.3.11 `n_instr*` `cree_n_instr_affect` (`n_var *` *var*, `n_exp *` *exp*)

4.10.3.12 `n_instr*` `cree_n_instr_appel` (`n_appel *` *appel*)

4.10.3.13 `n_instr*` `cree_n_instr_bloc` (`n_l_instr *` *liste*)

4.10.3.14 `n_instr*` `cree_n_instr_ecrire` (`n_exp *` *expression*)

4.10.3.15 `n_instr* cree_n_instr_faire (n_instr * faire, n_exp * test)`

4.10.3.16 `n_instr* cree_n_instr_incr (n_exp * incr)`

4.10.3.17 `n_instr* cree_n_instr_pour (n_instr * init, n_exp * test, n_instr * incr, n_instr * faire)`

4.10.3.18 `n_instr* cree_n_instr_retour (n_exp * expression)`

4.10.3.19 `n_instr* cree_n_instr_si (n_exp * test, n_instr * alors, n_instr * sinon)`

4.10.3.20 `n_instr* cree_n_instr_tantque (n_exp * test, n_instr * faire)`

4.10.3.21 `n_instr* cree_n_instr_vide (void)`

4.10.3.22 `n_l_dec* cree_n_l_dec (n_dec * tete, n_l_dec * queue)`

4.10.3.23 `n_l_exp* cree_n_l_exp (n_exp * tete, n_l_exp * queue)`

4.10.3.24 `n_l_instr* cree_n_l_instr (n_instr * tete, n_l_instr * queue)`

4.10.3.25 `n_prog* cree_n_prog (n_l_dec * variables, n_l_dec * fonctions)`

4.10.3.26 `n_var* cree_n_var_indicee (char * nom, n_exp * indice)`

4.10.3.27 `n_var* cree_n_var_simple (char * nom)`

4.11 inc/util.h File Reference

Functions

- `char * duplique_chaine (char *s)`
- `void erreur (char *message)`
- `void erreur_1s (char *message, char *s)`
- `void affiche_balise_ouvrante (const char *fct_, int trace_xml)`
- `void affiche_balise_fermante (const char *fct_, int trace_xml)`
- `void affiche_element (char *fct_, char *texte_, int trace_xml)`
- `void affiche_texte (char *texte_, int trace_xml)`

4.11.1 Function Documentation

4.11.1.1 void affiche_balise_fermante (const char * *fct_*, int *trace_xml*)

4.11.1.2 void affiche_balise_ouvrante (const char * *fct_*, int *trace_xml*)

4.11.1.3 void affiche_element (char * *fct_*, char * *texte_*, int *trace_xml*)

4.11.1.4 void affiche_texte (char * *texte_*, int *trace_xml*)

4.11.1.5 char* duplique_chaine (char * *s*)

4.11.1.6 void erreur (char * *message*)

4.11.1.7 void erreur_1s (char * *message*, char * *s*)

4.12 src/affiche_arbre_abstrait.c File Reference

```
#include <stdio.h>
#include "syntabs.h"
#include "util.h"
```

Functions

- void affiche_n_prog (n_prog *n)
- void affiche_l_instr (n_l_instr *n)
- void affiche_instr (n_instr *n)
- void affiche_instr_si (n_instr *n)
- void affiche_instr_tantque (n_instr *n)
- void affiche_instr_faire (n_instr *n)
- void affiche_instr_pour (n_instr *n)
- void affiche_instr_affect (n_instr *n)
- void affiche_instr_appel (n_instr *n)
- void affiche_instr_retour (n_instr *n)
- void affiche_instr_ecrire (n_instr *n)
- void affiche_l_exp (n_l_exp *n)
- void affiche_exp (n_exp *n)
- void affiche_varExp (n_exp *n)
- void affiche_opExp (n_exp *n)
- void affiche_intExp (n_exp *n)
- void affiche_lireExp (n_exp *n)
- void affiche_appelExp (n_exp *n)
- void affiche_l_dec (n_l_dec *n)
- void affiche_dec (n_dec *n)
- void affiche_foncDec (n_dec *n)
- void affiche_varDec (n_dec *n)
- void affiche_tabDec (n_dec *n)
- void affiche_var (n_var *n)
- void affiche_var_simple (n_var *n)
- void affiche_var_indicee (n_var *n)
- void affiche_appel (n_appel *n)

Variables

- int [trace_abs](#) = 1

4.12.1 Function Documentation

4.12.1.1 void affiche_appel (n_appel * n)

4.12.1.2 void affiche_appelExp (n_exp * n)

4.12.1.3 void affiche_dec (n_dec * n)

4.12.1.4 void affiche_exp (n_exp * n)

4.12.1.5 void affiche_foncDec (n_dec * n)

4.12.1.6 void affiche_instr (n_instr * n)

4.12.1.7 void affiche_instr_affect (n_instr * n)

4.12.1.8 void affiche_instr_appel (n_instr * n)

4.12.1.9 void affiche_instr_ecrire (n_instr * n)

4.12.1.10 void affiche_instr_faire (n_instr * n)

4.12.1.11 void affiche_instr_pour (n_instr * n)

4.12.1.12 void affiche_instr_retour (n_instr * n)

4.12.1.13 void affiche_instr_si (n_instr * n)

4.12.1.14 void affiche_instr_tantque (n_instr * n)

4.12.1.15 void affiche_intExp (n_exp * n)

4.12.1.16 void affiche_l_dec (n_l_dec * n)

4.12.1.17 void affiche_l_exp (n_l_exp * n)

4.12.1.18 void affiche_l_instr (n_l_instr * n)

4.12.1.19 void affiche_lireExp (n_exp * n)

4.12.1.20 void affiche_n_prog (n_prog * n)

4.12.1.21 void affiche_opExp (n_exp * n)

4.12.1.22 void affiche_tabDec (n_dec * n)

4.12.1.23 void affiche_var (n_var * n)

4.12.1.24 void affiche_var_indicee (n_var * n)

4.12.1.25 void affiche_var_simple (n_var * n)

4.12.1.26 void affiche_varDec (n_dec * n)

4.12.1.27 void affiche_varExp (n_exp * n)

4.12.2 Variable Documentation

4.12.2.1 int trace_abs = 1

4.13 src/analyseur_lexical.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "symboles.h"
#include "analyseur_lexical.h"
#include "util.h"
```

Macros

- #define YYTEXT_MAX 100
- #define is_num(c) (('0' <= (c)) && ((c) <= '9'))
- #define is_maj(c) (('A' <= (c)) && ((c) <= 'Z'))
- #define is_min(c) (('a' <= (c)) && ((c) <= 'z'))
- #define is_alpha(c) (is_maj(c) || is_min(c) || (c) == '_' || (c) == '\$')
- #define is_alphanum(c) (is_num((c)) || is_alpha((c)))

Functions

- int mangeEspaces ()
- char lireCar (void)
- void delireCar ()
- int yylex (void)
- void nom_token (int token, char *nom, char *valeur)
- void test_yylex_internal (FILE *yyin)

Variables

- FILE * `yyin`
- char * `tableMotsClefs` []
- int `codeMotClefs` []
- char `yytext` [YYTEXT_MAX]
- int `yyleng`
- int `nbMotsClefs` = 10
- int `motsClefsMaxLeng` = 7
- int `nb_ligne` = 1

4.13.1 Macro Definition Documentation

4.13.1.1 `#define is_alpha(c) (is_maj(c) || is_min(c) || (c) == '_' || (c) == '$')`

4.13.1.2 `#define is_alphanum(c) (is_num((c)) || is_alpha((c)))`

4.13.1.3 `#define is_maj(c) (('A' <= (c)) && ((c) <= 'Z'))`

4.13.1.4 `#define is_min(c) (('a' <= (c)) && ((c) <= 'z'))`

4.13.1.5 `#define is_num(c) (('0' <= (c)) && ((c) <= '9'))`

4.13.1.6 `#define YYTEXT_MAX 100`

4.13.2 Function Documentation

4.13.2.1 `void delireCar ()`

4.13.2.2 `char lireCar (void)`

4.13.2.3 `int mangeEspaces ()`

4.13.2.4 `void nom_token (int token, char * nom, char * valeur)`

4.13.2.5 `void test_yylex_internal (FILE * yyin)`

4.13.2.6 `int yylex (void)`

4.13.3 Variable Documentation

4.13.3.1 `int codeMotClefs[]`

Initial value:

```
= {
    SI, ALORS, SINON, TANTQUE, FAIRE, RETOUR, ENTIER,
    LIRE, ECRIRE, POUR
}
```


4.13.3.2 int motsClefsMaxLeng = 7

4.13.3.3 int nb_ligne = 1

4.13.3.4 int nbMotsClefs = 10

4.13.3.5 char* tableMotsClefs[]

Initial value:

```
= {
    "si", "alors", "sinon", "tantque", "faire", "retour", "entier", "lire", "ecrire", "pour"
}
```

4.13.3.6 FILE* yyin

4.13.3.7 int yyleng

4.13.3.8 char yytext[YYTEXT_MAX]

4.14 src/analyseur_semantique.c File Reference

```
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "analyseur_semantique.h"
#include "syntabs.h"
#include "util.h"
#include "dico.h"
```

Functions

- void [analyse_n_prog](#) (n_prog *n)
- void [analyse_l_instr](#) (n_l_instr *n)
- void [analyse_instr](#) (n_instr *n)
- void [analyse_instr_si](#) (n_instr *n)
- void [analyse_instr_tantque](#) (n_instr *n)
- void [analyse_instr_faire](#) (n_instr *n)
- void [analyse_instr_pour](#) (n_instr *n)
- void [analyse_instr_affect](#) (n_instr *n)
- void [analyse_instr_appel](#) (n_instr *n)
- void [analyse_instr_retour](#) (n_instr *n)
- void [analyse_instr_ecrire](#) (n_instr *n)
- void [analyse_l_exp](#) (n_l_exp *n)
- void [analyse_exp](#) (n_exp *n)
- void [analyse_varExp](#) (n_exp *n)
- void [analyse_opExp](#) (n_exp *n)

- void `analyse_intExp` (`n_exp *n`)
- void `analyse_lireExp` (`n_exp *n`)
- void `analyse_appelExp` (`n_exp *n`)
- void `analyse_l_dec` (`n_l_dec *n`)
- void `analyse_dec` (`n_dec *n`)
- void `analyse_foncDec` (`n_dec *n`)
- void `analyse_varDec` (`n_dec *n`)
- void `analyse_tabDec` (`n_dec *n`)
- void `analyse_var` (`n_var *n`, `char *s`)
- void `analyse_var_simple` (`n_var *n`, `char *s`)
- void `analyse_var_indicee` (`n_var *n`, `char *s`)
- void `analyse_appel` (`n_appel *n`)
- void `semantique` (`n_prog *p`, `int trace_dico`, `int print_mips`)

Main function to parse syntaxique tree.

- int `taille_n_l_dec` (`n_l_dec *liste`)
- int `taille_n_l_exp` (`n_l_exp *liste`)
- int `newEtiquette` (`void`)
- int `mips_print` (`const char *format,...`)
- void `mips_debut_fonction` ()
- void `mips_fin_fonction` ()
- void `mips_empile` (`char *s`)
- void `mips_depile` (`char *s`)

Variables

- int `contexte` = `C_VARIABLE_GLOBALE`
- int `adresseLocaleCourante` = 0
- int `adresseArgumentCourant` = 0
- int `trace_tab`
- int `trace_mips`
- int `nb_args_function`
- int `nb_var_local`
- int `etiquette` = 0
- int `asreturn`

4.14.1 Function Documentation

4.14.1.1 void `analyse_appel` (`n_appel * n`)

4.14.1.2 void `analyse_appelExp` (`n_exp * n`)

4.14.1.3 void `analyse_dec` (`n_dec * n`)

4.14.1.4 void `analyse_exp` (`n_exp * n`)

4.14.1.5 void `analyse_foncDec` (`n_dec * n`)

4.14.1.6 void `analyse_instr` (`n_instr * n`)

- 4.14.1.7 void analyse_instr_affect (n_instr * n)
- 4.14.1.8 void analyse_instr_appel (n_instr * n)
- 4.14.1.9 void analyse_instr_ecrire (n_instr * n)
- 4.14.1.10 void analyse_instr_faire (n_instr * n)
- 4.14.1.11 void analyse_instr_pour (n_instr * n)
- 4.14.1.12 void analyse_instr_retour (n_instr * n)
- 4.14.1.13 void analyse_instr_si (n_instr * n)
- 4.14.1.14 void analyse_instr_tantque (n_instr * n)
- 4.14.1.15 void analyse_intExp (n_exp * n)
- 4.14.1.16 void analyse_l_dec (n_l_dec * n)
- 4.14.1.17 void analyse_l_exp (n_l_exp * n)
- 4.14.1.18 void analyse_l_instr (n_l_instr * n)
- 4.14.1.19 void analyse_lireExp (n_exp * n)
- 4.14.1.20 void analyse_n_prog (n_prog * n)
- 4.14.1.21 void analyse_opExp (n_exp * n)
- 4.14.1.22 void analyse_tabDec (n_dec * n)
- 4.14.1.23 void analyse_var (n_var * n, char * s)
- 4.14.1.24 void analyse_var_indicee (n_var * n, char * s)
- 4.14.1.25 void analyse_var_simple (n_var * n, char * s)
- 4.14.1.26 void analyse_varDec (n_dec * n)
- 4.14.1.27 void analyse_varExp (n_exp * n)
- 4.14.1.28 void mips_debut_fonction ()
- 4.14.1.29 void mips_depile (char * s)
- 4.14.1.30 void mips_empile (char * s)
- 4.14.1.31 void mips_fin_function ()
- 4.14.1.32 int mips_print (const char * *format*, ...)
- 4.14.1.33 int newEtiquette (void)
- 4.14.1.34 void semantique (n_prog * p, int *trace_dico*, int *print_mips*)

Main function to parse syntaxique tree.

Parameters

	<i>p</i>	The head of the tree
in	<i>trace_dico</i>	boolean to display symbol table
in	<i>print_mips</i>	boolean to display the mips output on stdin

4.14.1.35 `int taille_n_l_dec (n_l_dec * liste)`

4.14.1.36 `int taille_n_l_exp (n_l_exp * liste)`

4.14.2 Variable Documentation

4.14.2.1 `int adresseArgumentCourant = 0`

4.14.2.2 `int adresseLocaleCourante = 0`

4.14.2.3 `int asreturn`

4.14.2.4 `int contexte = C_VARIABLE_GLOBALE`

4.14.2.5 `int etiquette = 0`

4.14.2.6 `int nb_args_function`

4.14.2.7 `int nb_var_local`

4.14.2.8 `int trace_mips`

4.14.2.9 `int trace_tab`

4.15 src/analyseur_syntaxique.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "symboles.h"
#include "analyseur_syntaxique.h"
#include "util.h"
#include "analyseur_lexical.h"
#include "premiers.h"
#include "suivants.h"
```

Functions

- void [EatTerminal](#) (void)
- void [DisplayErreur](#) (void)
- [n_l_dec](#) * [listeDecVariables](#) (void)
- [n_l_dec](#) * [listeDecFonctions](#) (void)
- [n_dec](#) * [declarationVariable](#) (void)
- [n_dec](#) * [declarationFonction](#) (void)
- [n_l_dec](#) * [listeParam](#) (void)
- [n_l_instr](#) * [listeInstructions](#) (void)
- [n_instr](#) * [instruction](#) (void)
- [n_instr](#) * [instructionAffect](#) (void)
- [n_instr](#) * [instructionBloc](#) (void)
- [n_instr](#) * [instructionSi](#) (void)
- [n_instr](#) * [instructionTantque](#) (void)
- [n_instr](#) * [instructionAppel](#) (void)
- [n_instr](#) * [instructionRetour](#) (void)
- [n_instr](#) * [instructionEcriture](#) (void)
- [n_instr](#) * [instructionVide](#) (void)
- [n_var](#) * [var](#) (void)
- [n_instr](#) * [instructionPour](#) (void)
- [n_exp](#) * [expression](#) (void)
- [n_appel](#) * [appelFct](#) (void)
- [n_exp](#) * [conjonction](#) (void)
- [n_exp](#) * [negation](#) (void)
- [n_exp](#) * [comparaison](#) (void)
- [n_exp](#) * [expArith](#) (void)
- [n_exp](#) * [terme](#) (void)
- [n_exp](#) * [facteur](#) (void)
- [n_l_exp](#) * [listeExpressions](#) (void)
- [n_l_exp](#) * [listeExpressionsBis](#) ([n_l_exp](#) *herite)
- [n_prog](#) * [programme](#) (void)
- [n_exp](#) * [conjonctionBis](#) ([n_exp](#) *herite)
- int [optTailleTableau](#) (void)
- [n_exp](#) * [expArithBis](#) ([n_exp](#) *herite)
- [n_instr](#) * [optSinon](#) (void)
- [n_exp](#) * [comparaisonBis](#) ([n_exp](#) *herite)
- [n_l_dec](#) * [optDecVariables](#) (void)
- [n_exp](#) * [optIndice](#) (void)
- [n_l_dec](#) * [listeDecVariablesBis](#) ()
- [n_exp](#) * [termeBis](#) ([n_exp](#) *herite)
- [n_exp](#) * [expressionBis](#) ([n_exp](#) *herite)
- [n_l_dec](#) * [optListeDecVariables](#) (void)
- [n_prog](#) * [syntaxe](#) (int trace_xml_tree)

Variables

- int [uniteCourante](#)
- int [trace_xml](#)
- char [nom](#) [100]
- char [valeur](#) [100]
- int [nb_ligne](#)
- char [yytext](#) []

4.15.1 Function Documentation

4.15.1.1 **n_appel*** appelFct (void)

4.15.1.2 **n_exp*** comparaison (void)

4.15.1.3 **n_exp*** comparaisonBis (**n_exp** * *herite*)

4.15.1.4 **n_exp*** conjonction (void)

4.15.1.5 **n_exp*** conjonctionBis (**n_exp** * *herite*)

4.15.1.6 **n_dec*** declarationFonction (void)

4.15.1.7 **n_dec*** declarationVariable (void)

4.15.1.8 void DisplayErreur (void)

4.15.1.9 void EatTerminal (void)

4.15.1.10 **n_exp*** expArith (void)

4.15.1.11 **n_exp*** expArithBis (**n_exp** * *herite*)

4.15.1.12 **n_exp*** expression (void)

4.15.1.13 **n_exp*** expressionBis (**n_exp** * *herite*)

4.15.1.14 **n_exp*** facteur (void)

4.15.1.15 **n_instr*** instruction (void)

4.15.1.16 **n_instr*** instructionAffect (void)

4.15.1.17 **n_instr*** instructionAppel (void)

4.15.1.18 **n_instr*** instructionBloc (void)

4.15.1.19 **n_instr*** instructionEcriture (void)

4.15.1.20 **n_instr*** instructionPour (void)

4.15.1.21 **n_instr*** instructionRetour (void)

4.15.1.22 **n_instr*** instructionSi (void)

- 4.15.1.23 `n_instr*` `instructionTantque (void)`
- 4.15.1.24 `n_instr*` `instructionVide (void)`
- 4.15.1.25 `n_l_dec*` `listeDecFonctions (void)`
- 4.15.1.26 `n_l_dec*` `listeDecVariables (void)`
- 4.15.1.27 `n_l_dec*` `listeDecVariablesBis ()`
- 4.15.1.28 `n_l_exp*` `listeExpressions (void)`
- 4.15.1.29 `n_l_exp*` `listeExpressionsBis (n_l_exp * herite)`
- 4.15.1.30 `n_l_instr*` `listeInstructions (void)`
- 4.15.1.31 `n_l_dec*` `listeParam (void)`
- 4.15.1.32 `n_exp*` `negation (void)`
- 4.15.1.33 `n_l_dec*` `optDecVariables (void)`
- 4.15.1.34 `n_exp*` `optIndice (void)`
- 4.15.1.35 `n_l_dec*` `optListeDecVariables (void)`
- 4.15.1.36 `n_instr*` `optSinon (void)`
- 4.15.1.37 `int` `optTailleTableau (void)`
- 4.15.1.38 `n_prog*` `programme (void)`
- 4.15.1.39 `n_prog*` `syntaxe (int trace_xml_tree)`
- 4.15.1.40 `n_exp*` `terme (void)`
- 4.15.1.41 `n_exp*` `termeBis (n_exp * herite)`
- 4.15.1.42 `n_var*` `var (void)`

4.15.2 Variable Documentation

4.15.2.1 `int` `nb_ligne`

4.15.2.2 `char` `nom[100]`

4.15.2.3 `int trace_xml`

4.15.2.4 `int uniteCourante`

4.15.2.5 `char valeur[100]`

4.15.2.6 `char yytext[]`

4.16 `src/compilateur.c` File Reference

```
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "affiche_arbre_abstrait.h"
#include "analyseur_lexical.h"
#include "analyseur_syntaxique.h"
#include "analyseur_semantique.h"
#include "freedom.h"
```

Functions

- `int main` (`int argc`, `char **argv`)

Variables

- `char yytext` [100]
- `FILE * yyin`

4.16.1 Function Documentation

4.16.1.1 `int main (int argc, char ** argv)`

4.16.2 Variable Documentation

4.16.2.1 `FILE* yyin`

4.16.2.2 `char yytext[100]`

4.17 `src/dico.c` File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "dico.h"
```


Functions

- void [entreeFonction](#) (void)
- void [sortieFonction](#) (void)
- int [ajoutIdentificateur](#) (char *identif, int classe, int type, int adresse, int complement)
- int [rechercheExecutable](#) (char *identif)
- int [rechercheDeclarative](#) (char *identif)
- void [affiche_dico](#) (void)

4.17.1 Function Documentation

4.17.1.1 void [affiche_dico](#) (void)

4.17.1.2 int [ajoutIdentificateur](#) (char * *identif*, int *classe*, int *type*, int *adresse*, int *complement*)

4.17.1.3 void [entreeFonction](#) (void)

4.17.1.4 int [rechercheDeclarative](#) (char * *identif*)

4.17.1.5 int [rechercheExecutable](#) (char * *identif*)

4.17.1.6 void [sortieFonction](#) (void)

4.18 src/freedom.c File Reference

```
#include <stdlib.h>
#include "syntabs.h"
#include "dico.h"
```

Functions

- void [free_dico](#) ()
- void [free_n_prog](#) (n_prog *n)
- void [free_l_instr](#) (n_l_instr *n)
- void [free_instr](#) (n_instr *n)
- void [free_instr_si](#) (n_instr *n)
- void [free_instr_tantque](#) (n_instr *n)
- void [free_instr_faire](#) (n_instr *n)
- void [free_instr_pour](#) (n_instr *n)
- void [free_instr_affect](#) (n_instr *n)
- void [free_instr_appel](#) (n_instr *n)
- void [free_instr_retour](#) (n_instr *n)
- void [free_instr_ecrire](#) (n_instr *n)
- void [free_l_exp](#) (n_l_exp *n)
- void [free_exp](#) (n_exp *n)
- void [free_varExp](#) (n_exp *n)
- void [free_opExp](#) (n_exp *n)
- void [free_intExp](#) (n_exp *n)

- void [free_lireExp](#) (n_exp *n)
- void [free_appelExp](#) (n_exp *n)
- void [free_l_dec](#) (n_l_dec *n)
- void [free_dec](#) (n_dec *n)
- void [free_foncDec](#) (n_dec *n)
- void [free_varDec](#) (n_dec *n)
- void [free_tabDec](#) (n_dec *n)
- void [free_var](#) (n_var *n)
- void [free_var_simple](#) (n_var *n)
- void [free_var_indicee](#) (n_var *n)
- void [free_appel](#) (n_appel *n)
- void [freedom](#) (n_prog *n)

4.18.1 Function Documentation

4.18.1.1 void [free_appel](#) (n_appel * n)

4.18.1.2 void [free_appelExp](#) (n_exp * n)

4.18.1.3 void [free_dec](#) (n_dec * n)

4.18.1.4 void [free_dico](#) ()

4.18.1.5 void [free_exp](#) (n_exp * n)

4.18.1.6 void [free_foncDec](#) (n_dec * n)

4.18.1.7 void [free_instr](#) (n_instr * n)

4.18.1.8 void [free_instr_affect](#) (n_instr * n)

4.18.1.9 void [free_instr_appel](#) (n_instr * n)

4.18.1.10 void [free_instr_ecrire](#) (n_instr * n)

4.18.1.11 void [free_instr_faire](#) (n_instr * n)

4.18.1.12 void [free_instr_pour](#) (n_instr * n)

4.18.1.13 void [free_instr_retour](#) (n_instr * n)

4.18.1.14 void [free_instr_si](#) (n_instr * n)

4.18.1.15 void [free_instr_tantque](#) (n_instr * n)

4.18.1.16 void [free_intExp](#) (n_exp * n)

4.18.1.17 void free_l_dec (n_l_dec * n)

4.18.1.18 void free_l_exp (n_l_exp * n)

4.18.1.19 void free_l_instr (n_l_instr * n)

4.18.1.20 void free_lireExp (n_exp * n)

4.18.1.21 void free_n_prog (n_prog * n)

4.18.1.22 void free_opExp (n_exp * n)

4.18.1.23 void free_tabDec (n_dec * n)

4.18.1.24 void free_var (n_var * n)

4.18.1.25 void free_var_indicee (n_var * n)

4.18.1.26 void free_var_simple (n_var * n)

4.18.1.27 void free_varDec (n_dec * n)

4.18.1.28 void free_varExp (n_exp * n)

4.18.1.29 void freedom (n_prog * n)

4.19 src/premiers.c File Reference

```
#include "symboles.h"
#include "premiers.h"
```

Functions

- void [initialise_premiers](#) (void)
- int [est_premier](#) (int terminal, int non_terminal)

4.19.1 Function Documentation

4.19.1.1 int [est_premier](#) (int *terminal*, int *non_terminal*)

4.19.1.2 void [initialise_premiers](#) (void)

4.20 src/suivants.c File Reference

```
#include "symboles.h"
#include "suivants.h"
```

Functions

- void `initialise_suivants` (void)
- int `est_suivant` (int terminal, int non_terminal)

4.20.1 Function Documentation

4.20.1.1 int `est_suivant` (int *terminal*, int *non_terminal*)

4.20.1.2 void `initialise_suivants` (void)

4.21 src/syntax.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "util.h"
#include "syntax.h"
```

Functions

- `n_appel` * `cree_n_appel` (char *fonction, `n_l_exp` *args)
- `n_prog` * `cree_n_prog` (`n_l_dec` *variables, `n_l_dec` *fonctions)
- `n_var` * `cree_n_var_simple` (char *nom)
- `n_var` * `cree_n_var_indicee` (char *nom, `n_exp` *indice)
- `n_exp` * `cree_n_exp_op` (operation op, `n_exp` *op1, `n_exp` *op2)
- `n_exp` * `cree_n_exp_appel` (`n_appel` *app)
- `n_exp` * `cree_n_exp_incr` (`n_var` *var)
- `n_exp` * `cree_n_exp_var` (`n_var` *var)
- `n_exp` * `cree_n_exp_entier` (int entier)
- `n_exp` * `cree_n_exp_lire` ()
- `n_l_exp` * `cree_n_l_exp` (`n_exp` *tete, `n_l_exp` *queue)
- `n_instr` * `cree_n_instr_incr` (`n_exp` *incr)
- `n_instr` * `cree_n_instr_si` (`n_exp` *test, `n_instr` *alors, `n_instr` *sinon)
- `n_instr` * `cree_n_instr_tantque` (`n_exp` *test, `n_instr` *faire)
- `n_instr` * `cree_n_instr_faire` (`n_instr` *faire, `n_exp` *test)
- `n_instr` * `cree_n_instr_pour` (`n_instr` *init, `n_exp` *test, `n_instr` *incr, `n_instr` *faire)
- `n_instr` * `cree_n_instr_affect` (`n_var` *var, `n_exp` *exp)
- `n_l_instr` * `cree_n_l_instr` (`n_instr` *tete, `n_l_instr` *queue)
- `n_instr` * `cree_n_instr_bloc` (`n_l_instr` *liste)
- `n_instr` * `cree_n_instr_appel` (`n_appel` *app)
- `n_instr` * `cree_n_instr_ecrire` (`n_exp` *expression)
- `n_instr` * `cree_n_instr_retour` (`n_exp` *expression)
- `n_instr` * `cree_n_instr_vide` (void)
- `n_dec` * `cree_n_dec_var` (char *nom)
- `n_dec` * `cree_n_dec_tab` (char *nom, int taille)
- `n_dec` * `cree_n_dec_fonc` (char *nom, `n_l_dec` *param, `n_l_dec` *variables, `n_instr` *corps)
- `n_l_dec` * `cree_n_l_dec` (`n_dec` *tete, `n_l_dec` *queue)

4.21.1 Function Documentation

4.21.1.1 `n_appel*` `cree_n_appel` (`char *` *fonction*, `n_l_exp *` *args*)

4.21.1.2 `n_dec*` `cree_n_dec_fonc` (`char *` *nom*, `n_l_dec *` *param*, `n_l_dec *` *variables*, `n_instr *` *corps*)

4.21.1.3 `n_dec*` `cree_n_dec_tab` (`char *` *nom*, `int` *taille*)

4.21.1.4 `n_dec*` `cree_n_dec_var` (`char *` *nom*)

4.21.1.5 `n_exp*` `cree_n_exp_appel` (`n_appel *` *app*)

4.21.1.6 `n_exp*` `cree_n_exp_entier` (`int` *entier*)

4.21.1.7 `n_exp*` `cree_n_exp_incr` (`n_var *` *var*)

4.21.1.8 `n_exp*` `cree_n_exp_lire` (`void`)

4.21.1.9 `n_exp*` `cree_n_exp_op` (`operation` *op*, `n_exp *` *op1*, `n_exp *` *op2*)

4.21.1.10 `n_exp*` `cree_n_exp_var` (`n_var *` *var*)

4.21.1.11 `n_instr*` `cree_n_instr_affect` (`n_var *` *var*, `n_exp *` *exp*)

4.21.1.12 `n_instr*` `cree_n_instr_appel` (`n_appel *` *app*)

4.21.1.13 `n_instr*` `cree_n_instr_bloc` (`n_l_instr *` *liste*)

4.21.1.14 `n_instr*` `cree_n_instr_ecrire` (`n_exp *` *expression*)

4.21.1.15 `n_instr*` `cree_n_instr_faire` (`n_instr *` *faire*, `n_exp *` *test*)

4.21.1.16 `n_instr*` `cree_n_instr_incr` (`n_exp *` *incr*)

4.21.1.17 `n_instr*` `cree_n_instr_pour` (`n_instr *` *init*, `n_exp *` *test*, `n_instr *` *incr*, `n_instr *` *faire*)

4.21.1.18 `n_instr*` `cree_n_instr_retour` (`n_exp *` *expression*)

4.21.1.19 `n_instr*` `cree_n_instr_si` (`n_exp *` *test*, `n_instr *` *alors*, `n_instr *` *sinon*)

4.21.1.20 `n_instr*` `cree_n_instr_tantque` (`n_exp *` *test*, `n_instr *` *faire*)

4.21.1.21 `n_instr*` `cree_n_instr_vide` (`void`)

4.21.1.22 `n_l_dec*` `cree_n_l_dec` (`n_dec *` *tete*, `n_l_dec *` *queue*)

4.21.1.23 `n_l_exp* cree_n_l_exp (n_exp * tete, n_l_exp * queue)`

4.21.1.24 `n_l_instr* cree_n_l_instr (n_instr * tete, n_l_instr * queue)`

4.21.1.25 `n_prog* cree_n_prog (n_l_dec * variables, n_l_dec * fonctions)`

4.21.1.26 `n_var* cree_n_var_indicee (char * nom, n_exp * indice)`

4.21.1.27 `n_var* cree_n_var_simple (char * nom)`

4.22 src/util.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Functions

- void `erreur` (char *message)
- void `erreur_1s` (char *message, char *s)
- char * `duplique_chaine` (char *src)
- void `indent` ()
- void `affiche_balise_ouvrante` (const char *fct_, int trace_xml)
- void `affiche_balise_fermante` (const char *fct_, int trace_xml)
- void `affiche_texte` (char *texte_, int trace_xml)
- void `affiche_xml_texte` (char *texte_)
- void `affiche_element` (char *fct_, char *texte_, int trace_xml)

Variables

- int `nb_ligne`
- int `indent_xml` = 0
- int `indent_step` = 1

4.22.1 Function Documentation

4.22.1.1 `void affiche_balise_fermante (const char * fct_, int trace_xml)`

4.22.1.2 `void affiche_balise_ouvrante (const char * fct_, int trace_xml)`

4.22.1.3 `void affiche_element (char * fct_, char * texte_, int trace_xml)`

4.22.1.4 `void affiche_texte (char * texte_, int trace_xml)`

4.22.1.5 `void affiche_xml_texte (char * texte_)`

4.22.1.6 char* duplique_chaine (char * *src*)

4.22.1.7 void erreur (char * *message*)

4.22.1.8 void erreur_1s (char * *message*, char * *s*)

4.22.1.9 void indent ()

4.22.2 Variable Documentation

4.22.2.1 int indent_step = 1

4.22.2.2 int indent_xml = 0

4.22.2.3 int nb_ligne

Index

- `__FREEDOM__`
 - `freedom.h`, 21
 - `_appelFct_`
 - `symboles.h`, 24
 - `_argumentsEffectifs_`
 - `symboles.h`, 24
 - `_comparaisonBis_`
 - `symboles.h`, 24
 - `_comparaison_`
 - `symboles.h`, 24
 - `_conjonctionBis_`
 - `symboles.h`, 24
 - `_conjonction_`
 - `symboles.h`, 24
 - `_declarationFonction_`
 - `symboles.h`, 24
 - `_declarationVariable_`
 - `symboles.h`, 24
 - `_expArithBis_`
 - `symboles.h`, 24
 - `_expArith_`
 - `symboles.h`, 24
 - `_expressionBis_`
 - `symboles.h`, 24
 - `_expression_`
 - `symboles.h`, 24
 - `_facteur_`
 - `symboles.h`, 24
 - `_instructionAffect_`
 - `symboles.h`, 24
 - `_instructionAppel_`
 - `symboles.h`, 24
 - `_instructionBloc_`
 - `symboles.h`, 24
 - `_instructionEcriture_`
 - `symboles.h`, 24
 - `_instructionFaire_`
 - `symboles.h`, 24
 - `_instructionPour_`
 - `symboles.h`, 24
 - `_instructionRetour_`
 - `symboles.h`, 24
 - `_instructionSi_`
 - `symboles.h`, 24
 - `_instructionTantque_`
 - `symboles.h`, 24
 - `_instructionVide_`
 - `symboles.h`, 25
 - `_instruction_`
 - `symboles.h`, 24
 - `_listeDecFonctions_`
 - `symboles.h`, 25
 - `_listeDecVariablesBis_`
 - `symboles.h`, 25
 - `_listeDecVariables_`
 - `symboles.h`, 25
 - `_listeExpressionsBis_`
 - `symboles.h`, 25
 - `_listeExpressions_`
 - `symboles.h`, 25
 - `_listeInstructions_`
 - `symboles.h`, 25
 - `_listeParam_`
 - `symboles.h`, 25
 - `_negation_`
 - `symboles.h`, 25
 - `_optDecVariables_`
 - `symboles.h`, 25
 - `_optIndice_`
 - `symboles.h`, 25
 - `_optListeDecVariables_`
 - `symboles.h`, 25
 - `_optSinon_`
 - `symboles.h`, 25
 - `_optTailleTableau_`
 - `symboles.h`, 25
 - `_programme_`
 - `symboles.h`, 25
 - `_termeBis_`
 - `symboles.h`, 25
 - `_terme_`
 - `symboles.h`, 25
 - `_var_`
 - `symboles.h`, 25
-
- `ACCOLADE_FERMANTE`
 - `symboles.h`, 25
 - `ACCOLADE_OUVRANTE`
 - `symboles.h`, 25
 - `ALORS`
 - `symboles.h`, 25
 - `adresse`
 - `desc_identif`, 5
 - `adresseArgumentCourant`
 - `analyseur_semantique.c`, 38
 - `dico.h`, 20
 - `adresseLocaleCourante`
 - `analyseur_semantique.c`, 38
 - `dico.h`, 20

- affecte_
 - n_instr_, 11
- affecteInst
 - n_instr_, 10
- affiche_appel
 - affiche_arbre_abstrait.c, 32
- affiche_appelExp
 - affiche_arbre_abstrait.c, 32
- affiche_arbre_abstrait.c
 - affiche_appel, 32
 - affiche_appelExp, 32
 - affiche_dec, 32
 - affiche_exp, 32
 - affiche_foncDec, 32
 - affiche_instr, 32
 - affiche_instr_affect, 32
 - affiche_instr_appel, 32
 - affiche_instr_ecrire, 32
 - affiche_instr_faire, 32
 - affiche_instr_pour, 32
 - affiche_instr_retour, 32
 - affiche_instr_si, 32
 - affiche_instr_tantque, 32
 - affiche_intExp, 32
 - affiche_l_dec, 32
 - affiche_l_exp, 32
 - affiche_l_instr, 32
 - affiche_lireExp, 32
 - affiche_n_prog, 32
 - affiche_opExp, 32
 - affiche_tabDec, 33
 - affiche_var, 33
 - affiche_var_indicee, 33
 - affiche_var_simple, 33
 - affiche_varDec, 33
 - affiche_varExp, 33
 - trace_abs, 33
- affiche_arbre_abstrait.h
 - affiche_n_prog, 15
- affiche_balise_fermante
 - util.c, 48
 - util.h, 31
- affiche_balise_ouvrante
 - util.c, 48
 - util.h, 31
- affiche_dec
 - affiche_arbre_abstrait.c, 32
- affiche_dico
 - dico.c, 43
 - dico.h, 20
- affiche_element
 - util.c, 48
 - util.h, 31
- affiche_exp
 - affiche_arbre_abstrait.c, 32
- affiche_foncDec
 - affiche_arbre_abstrait.c, 32
- affiche_instr
 - affiche_arbre_abstrait.c, 32
- affiche_instr_affect
 - affiche_arbre_abstrait.c, 32
- affiche_instr_appel
 - affiche_arbre_abstrait.c, 32
- affiche_instr_ecrire
 - affiche_arbre_abstrait.c, 32
- affiche_instr_faire
 - affiche_arbre_abstrait.c, 32
- affiche_instr_pour
 - affiche_arbre_abstrait.c, 32
- affiche_instr_retour
 - affiche_arbre_abstrait.c, 32
- affiche_instr_si
 - affiche_arbre_abstrait.c, 32
- affiche_instr_tantque
 - affiche_arbre_abstrait.c, 32
- affiche_intExp
 - affiche_arbre_abstrait.c, 32
- affiche_l_dec
 - affiche_arbre_abstrait.c, 32
- affiche_l_exp
 - affiche_arbre_abstrait.c, 32
- affiche_l_instr
 - affiche_arbre_abstrait.c, 32
- affiche_lireExp
 - affiche_arbre_abstrait.c, 32
- affiche_n_prog
 - affiche_arbre_abstrait.c, 32
 - affiche_arbre_abstrait.h, 15
- affiche_opExp
 - affiche_arbre_abstrait.c, 32
- affiche_tabDec
 - affiche_arbre_abstrait.c, 33
- affiche_texte
 - util.c, 48
 - util.h, 31
- affiche_var
 - affiche_arbre_abstrait.c, 33
- affiche_var_indicee
 - affiche_arbre_abstrait.c, 33
- affiche_var_simple
 - affiche_arbre_abstrait.c, 33
- affiche_varDec
 - affiche_arbre_abstrait.c, 33
- affiche_varExp
 - affiche_arbre_abstrait.c, 33
- affiche_xml_texte
 - util.c, 48
- ajoutIdentificateur
 - dico.c, 43
 - dico.h, 20
- alors
 - n_instr_, 11
- analyse_appel
 - analyseur_semantique.c, 36
- analyse_appelExp
 - analyseur_semantique.c, 36

- analyse_dec
 - analyseur_semantique.c, 36
- analyse_exp
 - analyseur_semantique.c, 36
- analyse_foncDec
 - analyseur_semantique.c, 36
- analyse_instr
 - analyseur_semantique.c, 36
- analyse_instr_affect
 - analyseur_semantique.c, 36
- analyse_instr_appel
 - analyseur_semantique.c, 37
- analyse_instr_ecrire
 - analyseur_semantique.c, 37
- analyse_instr_faire
 - analyseur_semantique.c, 37
- analyse_instr_pour
 - analyseur_semantique.c, 37
- analyse_instr_retour
 - analyseur_semantique.c, 37
- analyse_instr_si
 - analyseur_semantique.c, 37
- analyse_instr_tantque
 - analyseur_semantique.c, 37
- analyse_intExp
 - analyseur_semantique.c, 37
- analyse_l_dec
 - analyseur_semantique.c, 37
- analyse_l_exp
 - analyseur_semantique.c, 37
- analyse_l_instr
 - analyseur_semantique.c, 37
- analyse_lireExp
 - analyseur_semantique.c, 37
- analyse_n_prog
 - analyseur_semantique.c, 37
- analyse_opExp
 - analyseur_semantique.c, 37
- analyse_tabDec
 - analyseur_semantique.c, 37
- analyse_var
 - analyseur_semantique.c, 37
- analyse_var_indicee
 - analyseur_semantique.c, 37
- analyse_var_simple
 - analyseur_semantique.c, 37
- analyse_varDec
 - analyseur_semantique.c, 37
- analyse_varExp
 - analyseur_semantique.c, 37
- analyseur_lexical.c
 - codeMotClefs, 34
 - delireCar, 34
 - is_alpha, 34
 - is_alphanum, 34
 - is_maj, 34
 - is_min, 34
 - is_num, 34
 - lireCar, 34
 - mangeEspaces, 34
 - motsClefsMaxLeng, 34
 - nb_ligne, 35
 - nbMotsClefs, 35
 - nom_token, 34
 - tableMotsClefs, 35
 - test_yylex_internal, 34
 - YYTEXT_MAX, 34
 - yyin, 35
 - yylen, 35
 - yylex, 34
 - yytext, 35
- analyseur_lexical.h
 - nom_token, 15
 - test_yylex_internal, 15
 - yylex, 15
- analyseur_semantique.c
 - adresseArgumentCourant, 38
 - adresseLocaleCourante, 38
 - analyse_appel, 36
 - analyse_appelExp, 36
 - analyse_dec, 36
 - analyse_exp, 36
 - analyse_foncDec, 36
 - analyse_instr, 36
 - analyse_instr_affect, 36
 - analyse_instr_appel, 37
 - analyse_instr_ecrire, 37
 - analyse_instr_faire, 37
 - analyse_instr_pour, 37
 - analyse_instr_retour, 37
 - analyse_instr_si, 37
 - analyse_instr_tantque, 37
 - analyse_intExp, 37
 - analyse_l_dec, 37
 - analyse_l_exp, 37
 - analyse_l_instr, 37
 - analyse_lireExp, 37
 - analyse_n_prog, 37
 - analyse_opExp, 37
 - analyse_tabDec, 37
 - analyse_var, 37
 - analyse_var_indicee, 37
 - analyse_var_simple, 37
 - analyse_varDec, 37
 - analyse_varExp, 37
 - asreturn, 38
 - contexte, 38
 - etiquette, 38
 - mips_debut_fonction, 37
 - mips_depile, 37
 - mips_empile, 37
 - mips_fin_fonction, 37
 - mips_print, 37
 - nb_args_function, 38
 - nb_var_local, 38
 - newEtiquette, 37

- semantique, 37
- taille_n_l_dec, 38
- taille_n_l_exp, 38
- trace_mips, 38
- trace_tab, 38
- analyseur_semantique.h
 - semantique, 16
- analyseur_syntaxique.c
 - appelFct, 40
 - comparaison, 40
 - comparaisonBis, 40
 - conjonction, 40
 - conjonctionBis, 40
 - declarationFonction, 40
 - declarationVariable, 40
 - DisplayErreur, 40
 - EatTerminal, 40
 - expArith, 40
 - expArithBis, 40
 - expression, 40
 - expressionBis, 40
 - facteur, 40
 - instruction, 40
 - instructionAffect, 40
 - instructionAppel, 40
 - instructionBloc, 40
 - instructionEcriture, 40
 - instructionPour, 40
 - instructionRetour, 40
 - instructionSi, 40
 - instructionTantque, 40
 - instructionVide, 41
 - listeDecFonctions, 41
 - listeDecVariables, 41
 - listeDecVariablesBis, 41
 - listeExpressions, 41
 - listeExpressionsBis, 41
 - listeInstructions, 41
 - listeParam, 41
 - nb_ligne, 41
 - negation, 41
 - nom, 41
 - optDecVariables, 41
 - optIndice, 41
 - optListeDecVariables, 41
 - optSinon, 41
 - optTailleTableau, 41
 - programme, 41
 - syntaxe, 41
 - terme, 41
 - termeBis, 41
 - trace_xml, 41
 - uniteCourante, 42
 - valeur, 42
 - var, 41
 - yytext, 42
- analyseur_syntaxique.h
 - appelFct, 17
 - argumentsEffectifs, 17
 - comparaison, 17
 - comparaisonBis, 17
 - conjonction, 17
 - conjonctionBis, 17
 - declarationFonction, 17
 - declarationVariable, 17
 - DisplayErreur, 17
 - EatTerminal, 17
 - expArith, 17
 - expArithBis, 17
 - expression, 17
 - expressionBis, 17
 - facteur, 18
 - instruction, 18
 - instructionAffect, 18
 - instructionAppel, 18
 - instructionBloc, 18
 - instructionEcriture, 18
 - instructionFaire, 18
 - instructionPour, 18
 - instructionRetour, 18
 - instructionSi, 18
 - instructionTantque, 18
 - instructionVide, 18
 - listeDecFonctions, 18
 - listeDecVariables, 18
 - listeDecVariablesBis, 18
 - listeExpressions, 18
 - listeExpressionsBis, 18
 - listeInstructions, 18
 - listeParam, 18
 - negation, 18
 - optDecVariables, 18
 - optIndice, 18
 - optListeDecVariables, 18
 - optSinon, 19
 - optTailleTableau, 19
 - programme, 19
 - syntaxe, 19
 - terme, 19
 - termeBis, 19
 - var, 19
- appel
 - n_exp_, 9
 - n_instr_, 11
- appelExp
 - n_exp_, 8
- appelFct
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17
- appelInst
 - n_instr_, 10
- args
 - n_appel_, 6
- argumentsEffectifs
 - analyseur_syntaxique.h, 17
- asreturn

- analyseur_semantique.c, 38
- base
 - dico_, 6
- blocInst
 - n_instr_, 10
- C_ARGUMENT
 - dico.h, 20
- C_VARIABLE_GLOBALE
 - dico.h, 20
- C_VARIABLE_LOCALE
 - dico.h, 20
- CROCHET_FERMANT
 - symboles.h, 25
- CROCHET_OUVRANT
 - symboles.h, 25
- classe
 - desc_identif, 5
- codeMotClefs
 - analyseur_lexical.c, 34
- comparaison
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17
- comparaisonBis
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17
- compilateur.c
 - main, 42
 - yyin, 42
 - yytext, 42
- complement
 - desc_identif, 5
- conjonction
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17
- conjonctionBis
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17
- contexte
 - analyseur_semantique.c, 38
 - dico.h, 20
- corps
 - n_dec_, 7
- cree_n_appel
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_dec_fonc
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_dec_tab
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_dec_var
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_exp_appel
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_exp_entier
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_exp_incr
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_exp_lire
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_exp_op
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_exp_var
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_instr_affect
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_instr_appel
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_instr_bloc
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_instr_ecrire
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_instr_faire
 - syntabs.c, 47
 - syntabs.h, 29
- cree_n_instr_incr
 - syntabs.c, 47
 - syntabs.h, 30
- cree_n_instr_pour
 - syntabs.c, 47
 - syntabs.h, 30
- cree_n_instr_retour
 - syntabs.c, 47
 - syntabs.h, 30
- cree_n_instr_si
 - syntabs.c, 47
 - syntabs.h, 30
- cree_n_instr_tantque
 - syntabs.c, 47
 - syntabs.h, 30
- cree_n_instr_vide
 - syntabs.c, 47
 - syntabs.h, 30
- cree_n_l_dec
 - syntabs.c, 47
 - syntabs.h, 30
- cree_n_l_exp
 - syntabs.c, 47
 - syntabs.h, 30
- cree_n_l_instr
 - syntabs.c, 48
 - syntabs.h, 30
- cree_n_prog

- syntabs.c, 48
- syntabs.h, 30
- cree_n_var_indicee
 - syntabs.c, 48
 - syntabs.h, 30
- cree_n_var_simple
 - syntabs.c, 48
 - syntabs.h, 30
- DIVISE
 - symboles.h, 26
- declarationFonction
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17
- declarationVariable
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17
- delireCar
 - analyseur_lexical.c, 34
- desc_identif, 5
 - adresse, 5
 - classe, 5
 - complement, 5
 - identif, 5
 - type, 5
- dico
 - dico.h, 20
- dico.c
 - affiche_dico, 43
 - ajouteIdentificateur, 43
 - entreeFonction, 43
 - rechercheDeclarative, 43
 - rechercheExecutable, 43
 - sortieFonction, 43
- dico.h
 - adresseArgumentCourant, 20
 - adresseLocaleCourante, 20
 - affiche_dico, 20
 - ajouteIdentificateur, 20
 - C_ARGUMENT, 20
 - C_VARIABLE_GLOBALE, 20
 - C_VARIABLE_LOCALE, 20
 - contexte, 20
 - dico, 20
 - entreeFonction, 20
 - maxDico, 20
 - rechercheDeclarative, 20
 - rechercheExecutable, 20
 - sortieFonction, 20
 - T_ENTIER, 20
 - T_FONCTION, 20
 - T_TABLEAU_ENTIER, 20
- dico_, 5
 - base, 6
 - sommet, 6
 - tab, 6
- diff
 - syntabs.h, 29
- DisplayErreur
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17
- divise
 - syntabs.h, 29
- duplique_chaine
 - util.c, 48
 - util.h, 31
- ECRIRE
 - symboles.h, 26
- EGAL
 - symboles.h, 26
- ENTIER
 - symboles.h, 26
- EPSILON
 - symboles.h, 26
- EatTerminal
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17
- ecrire_
 - n_instr_, 11
- ecrireInst
 - n_instr_, 10
- egal
 - syntabs.h, 29
- entier
 - n_exp_, 9
- entreeFonction
 - dico.c, 43
 - dico.h, 20
- erreur
 - util.c, 49
 - util.h, 31
- erreur_1s
 - util.c, 49
 - util.h, 31
- est_premier
 - premiers.c, 45
 - premiers.h, 21
- est_suivant
 - suivants.c, 46
 - suivants.h, 22
- ET
 - symboles.h, 26
- et
 - syntabs.h, 29
- etiquette
 - analyseur_semantique.c, 38
- exp
 - n_instr_, 11
- expArith
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17
- expArithBis
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17
- expression
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17

- n_instr_, 11
- expressionBis
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 17
- FAIRE
 - symboles.h, 26
- FIN
 - symboles.h, 26
- FOIS
 - symboles.h, 26
- facteur
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 18
- faire
 - n_instr_, 11
- faire_
 - n_instr_, 11
- faireInst
 - n_instr_, 10
- fois
 - syntabs.h, 29
- foncDec
 - n_dec_, 7
- foncDec_
 - n_dec_, 7
- fonction
 - n_appel_, 6
- fonctions
 - n_prog_, 13
- free_appel
 - freedom.c, 44
- free_appelExp
 - freedom.c, 44
- free_dec
 - freedom.c, 44
- free_dico
 - freedom.c, 44
- free_exp
 - freedom.c, 44
- free_foncDec
 - freedom.c, 44
- free_instr
 - freedom.c, 44
- free_instr_affect
 - freedom.c, 44
- free_instr_appel
 - freedom.c, 44
- free_instr_ecrire
 - freedom.c, 44
- free_instr_faire
 - freedom.c, 44
- free_instr_pour
 - freedom.c, 44
- free_instr_retour
 - freedom.c, 44
- free_instr_si
 - freedom.c, 44
- free_instr_tantque
 - freedom.c, 44
- freedom.c, 44
- free_intExp
 - freedom.c, 44
- free_l_dec
 - freedom.c, 44
- free_l_exp
 - freedom.c, 45
- free_l_instr
 - freedom.c, 45
- free_lireExp
 - freedom.c, 45
- free_n_prog
 - freedom.c, 45
- free_opExp
 - freedom.c, 45
- free_tabDec
 - freedom.c, 45
- free_var
 - freedom.c, 45
- free_var_indicee
 - freedom.c, 45
- free_var_simple
 - freedom.c, 45
- free_varDec
 - freedom.c, 45
- free_varExp
 - freedom.c, 45
- freedom
 - freedom.c, 45
 - freedom.h, 21
- freedom.c
 - free_appel, 44
 - free_appelExp, 44
 - free_dec, 44
 - free_dico, 44
 - free_exp, 44
 - free_foncDec, 44
 - free_instr, 44
 - free_instr_affect, 44
 - free_instr_appel, 44
 - free_instr_ecrire, 44
 - free_instr_faire, 44
 - free_instr_pour, 44
 - free_instr_retour, 44
 - free_instr_si, 44
 - free_instr_tantque, 44
 - free_intExp, 44
 - free_l_dec, 44
 - free_l_exp, 45
 - free_l_instr, 45
 - free_lireExp, 45
 - free_n_prog, 45
 - free_opExp, 45
 - free_tabDec, 45
 - free_var, 45
 - free_var_indicee, 45
 - free_var_simple, 45
 - free_varDec, 45

- free_varExp, 45
 - freedom, 45
- freedom.h
 - __FREEDOM__, 21
 - freedom, 21
- ID_FCT
 - symboles.h, 26
- ID_VAR
 - symboles.h, 26
- INFERIEUR
 - symboles.h, 26
- identif
 - desc_identif, 5
- inc/affiche_arbre_abstrait.h, 15
- inc/analyseur_lexical.h, 15
- inc/analyseur_semantique.h, 15
- inc/analyseur_syntaxique.h, 16
- inc/dico.h, 19
- inc/freedom.h, 20
- inc/premiers.h, 21
- inc/suivants.h, 21
- inc/symboles.h, 22
- inc/syntabs.h, 27
- inc/util.h, 30
- incr
 - n_exp_, 9
 - n_instr_, 11
- incrExp
 - n_exp_, 8
- incrInst
 - n_instr_, 10
- indent
 - util.c, 49
- indent_step
 - util.c, 49
- indent_xml
 - util.c, 49
- indice
 - n_var_, 14
- indicee
 - n_var_, 14
- indicee_
 - n_var_, 14
- inf
 - syntabs.h, 29
- infeg
 - syntabs.h, 29
- init
 - n_instr_, 11
- initialise_premiers
 - premiers.c, 45
 - premiers.h, 21
- initialise_suivants
 - suivants.c, 46
 - suivants.h, 22
- instruction
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 18
- instructionAffect
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 18
- instructionAppel
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 18
- instructionBloc
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 18
- instructionEcriture
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 18
- instructionFaire
 - analyseur_syntaxique.h, 18
- instructionPour
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 18
- instructionRetour
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 18
- instructionSi
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 18
- instructionTantque
 - analyseur_syntaxique.c, 40
 - analyseur_syntaxique.h, 18
- instructionVide
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 18
- intExp
 - n_exp_, 8
- is_alpha
 - analyseur_lexical.c, 34
- is_alphanum
 - analyseur_lexical.c, 34
- is_maj
 - analyseur_lexical.c, 34
- is_min
 - analyseur_lexical.c, 34
- is_num
 - analyseur_lexical.c, 34
- LIRE
 - symboles.h, 26
- lireCar
 - analyseur_lexical.c, 34
- lireExp
 - n_exp_, 8
- liste
 - n_instr_, 11
- listeDecFonctions
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 18
- listeDecVariables
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 18
- listeDecVariablesBis
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 18

- listeExpressions
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 18
- listeExpressionsBis
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 18
- listeInstructions
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 18
- listeParam
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 18
- MOINS
 - symbles.h, 26
- main
 - compilateur.c, 42
- mangeEspaces
 - analyseur_lexical.c, 34
- maxDico
 - dico.h, 20
- mips_debut_fonction
 - analyseur_semantique.c, 37
- mips_depile
 - analyseur_semantique.c, 37
- mips_empile
 - analyseur_semantique.c, 37
- mips_fin_fonction
 - analyseur_semantique.c, 37
- mips_print
 - analyseur_semantique.c, 37
- modulo
 - syntabs.h, 29
- moins
 - syntabs.h, 28
- motsClefsMaxLeng
 - analyseur_lexical.c, 34
- n_appel
 - syntabs.h, 28
- n_appel_, 6
 - args, 6
 - fonction, 6
- n_dec
 - syntabs.h, 28
- n_dec_, 6
 - corps, 7
 - foncDec, 7
 - foncDec_, 7
 - nom, 7
 - param, 7
 - tabDec, 7
 - tabDec_, 7
 - taille, 7
 - type, 7
 - u, 8
 - varDec, 7
 - varDec_, 8
 - variables, 8
- n_exp
 - syntabs.h, 28
- n_exp_, 8
 - appel, 9
 - appelExp, 8
 - entier, 9
 - incr, 9
 - incrExp, 8
 - intExp, 8
 - lireExp, 8
 - op, 9
 - op1, 9
 - op2, 9
 - opExp, 8
 - opExp_, 9
 - type, 9
 - u, 9
 - var, 9
 - varExp, 8
- n_instr
 - syntabs.h, 28
- n_instr_, 9
 - affecte_, 11
 - affecteInst, 10
 - alors, 11
 - appel, 11
 - appelInst, 10
 - blocInst, 10
 - ecrire_, 11
 - ecrireInst, 10
 - exp, 11
 - expression, 11
 - faire, 11
 - faire_, 11
 - faireInst, 10
 - incr, 11
 - incrInst, 10
 - init, 11
 - liste, 11
 - pour_, 11
 - pourInst, 10
 - retour_, 11
 - retourInst, 10
 - si_, 11
 - siInst, 10
 - sinon, 11
 - tantque_, 11
 - tantqueInst, 10
 - test, 11
 - type, 11
 - u, 11
 - var, 11
 - videInst, 10
- n_l_dec
 - syntabs.h, 28
- n_l_dec_, 12
 - queue, 12
 - tete, 12

- n_l_exp
 - syntabs.h, 28
- n_l_exp_, 12
 - queue, 12
 - tete, 12
- n_l_instr
 - syntabs.h, 28
- n_l_instr_, 12
 - queue, 13
 - tete, 13
- n_prog
 - syntabs.h, 28
- n_prog_, 13
 - fonctions, 13
 - variables, 13
- n_var
 - syntabs.h, 28
- n_var_, 13
 - indice, 14
 - indicee, 14
 - indicee_, 14
 - nom, 14
 - simple, 14
 - type, 14
 - u, 14
- NB_NON_TERMINAUX
 - symboles.h, 26
- NB_TERMINAUX
 - symboles.h, 26
- NOMBRE
 - symboles.h, 26
- NON
 - symboles.h, 26
- nb_args_function
 - analyseur_semantique.c, 38
- nb_ligne
 - analyseur_lexical.c, 35
 - analyseur_syntaxique.c, 41
 - util.c, 49
- nb_var_local
 - analyseur_semantique.c, 38
- nbMotsClefs
 - analyseur_lexical.c, 35
- negatif
 - syntabs.h, 29
- negation
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 18
- newEtiquette
 - analyseur_semantique.c, 37
- nom
 - analyseur_syntaxique.c, 41
 - n_dec_, 7
 - n_var_, 14
- nom_token
 - analyseur_lexical.c, 34
 - analyseur_lexical.h, 15
- non
 - syntabs.h, 29
- op
 - n_exp_, 9
- op1
 - n_exp_, 9
- op2
 - n_exp_, 9
- opExp
 - n_exp_, 8
- opExp_
 - n_exp_, 9
- operation
 - syntabs.h, 28
- optDecVariables
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 18
- optIndice
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 18
- optListeDecVariables
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 18
- optSinon
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 19
- optTailleTableau
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 19
- OU
 - symboles.h, 26
- ou
 - syntabs.h, 29
- PARENTHESE_FERMANTE
 - symboles.h, 26
- PARENTHESE_OUVRANTE
 - symboles.h, 26
- PLUS
 - symboles.h, 26
- POINT_VIRGULE
 - symboles.h, 26
- POUR
 - symboles.h, 27
- param
 - n_dec_, 7
- plus
 - syntabs.h, 28
- pour_
 - n_instr_, 11
- pourInst
 - n_instr_, 10
- premiers
 - premiers.h, 21
- premiers.c
 - est_premier, 45
 - initialise_premiers, 45
- premiers.h
 - est_premier, 21

- initialise_premiers, 21
- premiers, 21
- programme
 - analyseur_syntaxique.c, 41
 - analyseur_syntaxique.h, 19
- queue
 - n_l_dec_, 12
 - n_l_exp_, 12
 - n_l_instr_, 13
- RETOUR
 - symboles.h, 27
- rechercheDeclarative
 - dico.c, 43
 - dico.h, 20
- rechercheExecutable
 - dico.c, 43
 - dico.h, 20
- retour_
 - n_instr_, 11
- retourInst
 - n_instr_, 10
- SINON
 - symboles.h, 27
- semantique
 - analyseur_semantique.c, 37
 - analyseur_semantique.h, 16
- SI
 - symboles.h, 27
- si_
 - n_instr_, 11
- siInst
 - n_instr_, 10
- simple
 - n_var_, 14
- sinon
 - n_instr_, 11
- sommet
 - dico_, 6
- sortieFonction
 - dico.c, 43
 - dico.h, 20
- src/affiche_arbre_abstrait.c, 31
- src/analyseur_lexical.c, 33
- src/analyseur_semantique.c, 35
- src/analyseur_syntaxique.c, 38
- src/compilateur.c, 42
- src/dico.c, 42
- src/freedom.c, 43
- src/premiers.c, 45
- src/suivants.c, 45
- src/syntax.h, 46
- src/util.c, 48
- suivants
 - suivants.h, 22
- suivants.c
 - est_suivant, 46
- initialise_suivants, 46
- suivants.h
 - est_suivant, 22
 - initialise_suivants, 22
 - suivants, 22
- sup
 - syntabs.h, 29
- supeg
 - syntabs.h, 29
- symboles.h
 - _appelFct_, 24
 - _argumentsEffectifs_, 24
 - _comparaisonBis_, 24
 - _comparaison_, 24
 - _conjonctionBis_, 24
 - _conjonction_, 24
 - _declarationFonction_, 24
 - _declarationVariable_, 24
 - _expArithBis_, 24
 - _expArith_, 24
 - _expressionBis_, 24
 - _expression_, 24
 - _facteur_, 24
 - _instructionAffect_, 24
 - _instructionAppel_, 24
 - _instructionBloc_, 24
 - _instructionEcriture_, 24
 - _instructionFaire_, 24
 - _instructionPour_, 24
 - _instructionRetour_, 24
 - _instructionSi_, 24
 - _instructionTantque_, 24
 - _instructionVide_, 25
 - _instruction_, 24
 - _listeDecFonctions_, 25
 - _listeDecVariablesBis_, 25
 - _listeDecVariables_, 25
 - _listeExpressionsBis_, 25
 - _listeExpressions_, 25
 - _listeInstructions_, 25
 - _listeParam_, 25
 - _negation_, 25
 - _optDecVariables_, 25
 - _optIndice_, 25
 - _optListeDecVariables_, 25
 - _optSinon_, 25
 - _optTailleTableau_, 25
 - _programme_, 25
 - _termeBis_, 25
 - _terme_, 25
 - _var_, 25
 - ACCOLADE_FERMANTE, 25
 - ACCOLADE_OUVRANTE, 25
 - ALORS, 25
 - CROCHET_FERMANT, 25
 - CROCHET_OUVRANT, 25
 - DIVISE, 26
 - ECRIRE, 26

- EGAL, [26](#)
- ENTIER, [26](#)
- EPSILON, [26](#)
- ET, [26](#)
- FAIRE, [26](#)
- FIN, [26](#)
- FOIS, [26](#)
- ID_FCT, [26](#)
- ID_VAR, [26](#)
- INFERIEUR, [26](#)
- LIRE, [26](#)
- MOINS, [26](#)
- NB_NON_TERMINAUX, [26](#)
- NB_TERMINAUX, [26](#)
- NOMBRE, [26](#)
- NON, [26](#)
- OU, [26](#)
- PARENTHESE_FERMANTE, [26](#)
- PARENTHESE_OUVRANTE, [26](#)
- PLUS, [26](#)
- POINT_VIRGULE, [26](#)
- POUR, [27](#)
- RETOUR, [27](#)
- SINON, [27](#)
- SI, [27](#)
- TANTQUE, [27](#)
- VIRGULE, [27](#)
- syntabs.c
 - cree_n_appel, [47](#)
 - cree_n_dec_fonc, [47](#)
 - cree_n_dec_tab, [47](#)
 - cree_n_dec_var, [47](#)
 - cree_n_exp_appel, [47](#)
 - cree_n_exp_entier, [47](#)
 - cree_n_exp_incr, [47](#)
 - cree_n_exp_lire, [47](#)
 - cree_n_exp_op, [47](#)
 - cree_n_exp_var, [47](#)
 - cree_n_instr_affect, [47](#)
 - cree_n_instr_appel, [47](#)
 - cree_n_instr_bloc, [47](#)
 - cree_n_instr_ecrire, [47](#)
 - cree_n_instr_faire, [47](#)
 - cree_n_instr_incr, [47](#)
 - cree_n_instr_pour, [47](#)
 - cree_n_instr_retour, [47](#)
 - cree_n_instr_si, [47](#)
 - cree_n_instr_tantque, [47](#)
 - cree_n_instr_vide, [47](#)
 - cree_n_l_dec, [47](#)
 - cree_n_l_exp, [47](#)
 - cree_n_l_instr, [48](#)
 - cree_n_prog, [48](#)
 - cree_n_var_indicee, [48](#)
 - cree_n_var_simple, [48](#)
- syntabs.h
 - cree_n_appel, [29](#)
 - cree_n_dec_fonc, [29](#)
 - cree_n_dec_tab, [29](#)
 - cree_n_dec_var, [29](#)
 - cree_n_exp_appel, [29](#)
 - cree_n_exp_entier, [29](#)
 - cree_n_exp_incr, [29](#)
 - cree_n_exp_lire, [29](#)
 - cree_n_exp_op, [29](#)
 - cree_n_exp_var, [29](#)
 - cree_n_instr_affect, [29](#)
 - cree_n_instr_appel, [29](#)
 - cree_n_instr_bloc, [29](#)
 - cree_n_instr_ecrire, [29](#)
 - cree_n_instr_faire, [29](#)
 - cree_n_instr_incr, [30](#)
 - cree_n_instr_pour, [30](#)
 - cree_n_instr_retour, [30](#)
 - cree_n_instr_si, [30](#)
 - cree_n_instr_tantque, [30](#)
 - cree_n_instr_vide, [30](#)
 - cree_n_l_dec, [30](#)
 - cree_n_l_exp, [30](#)
 - cree_n_l_instr, [30](#)
 - cree_n_prog, [30](#)
 - cree_n_var_indicee, [30](#)
 - cree_n_var_simple, [30](#)
 - diff, [29](#)
 - divise, [29](#)
 - egal, [29](#)
 - et, [29](#)
 - fois, [29](#)
 - inf, [29](#)
 - infeg, [29](#)
 - modulo, [29](#)
 - moins, [28](#)
 - n_appel, [28](#)
 - n_dec, [28](#)
 - n_exp, [28](#)
 - n_instr, [28](#)
 - n_l_dec, [28](#)
 - n_l_exp, [28](#)
 - n_l_instr, [28](#)
 - n_prog, [28](#)
 - n_var, [28](#)
 - negatif, [29](#)
 - non, [29](#)
 - operation, [28](#)
 - ou, [29](#)
 - plus, [28](#)
 - sup, [29](#)
 - supeg, [29](#)
- syntaxe
 - analyseur_syntaxique.c, [41](#)
 - analyseur_syntaxique.h, [19](#)
- T_ENTIER
 - dico.h, [20](#)
- T_FONCTION
 - dico.h, [20](#)
- T_TABLEAU_ENTIER

- dico.h, 20
- TANTQUE
 - symboles.h, 27
- tab
 - dico_, 6
- tabDec
 - n_dec_, 7
- tabDec_
 - n_dec_, 7
- tableMotsClefs
 - analysteur_lexical.c, 35
- taille
 - n_dec_, 7
- taille_n_l_dec
 - analysteur_semantique.c, 38
- taille_n_l_exp
 - analysteur_semantique.c, 38
- tantque_
 - n_instr_, 11
- tantqueInst
 - n_instr_, 10
- terme
 - analysteur_syntaxique.c, 41
 - analysteur_syntaxique.h, 19
- termeBis
 - analysteur_syntaxique.c, 41
 - analysteur_syntaxique.h, 19
- test
 - n_instr_, 11
- test_yylex_internal
 - analysteur_lexical.c, 34
 - analysteur_lexical.h, 15
- tete
 - n_l_dec_, 12
 - n_l_exp_, 12
 - n_l_instr_, 13
- trace_abs
 - affiche_arbre_abstrait.c, 33
- trace_mips
 - analysteur_semantique.c, 38
- trace_tab
 - analysteur_semantique.c, 38
- trace_xml
 - analysteur_syntaxique.c, 41
- type
 - desc_identif, 5
 - n_dec_, 7
 - n_exp_, 9
 - n_instr_, 11
 - n_var_, 14
- u
 - n_dec_, 8
 - n_exp_, 9
 - n_instr_, 11
 - n_var_, 14
- uniteCourante
 - analysteur_syntaxique.c, 42
- util.c
 - affiche_balise_fermante, 48
 - affiche_baliseouvrante, 48
 - affiche_element, 48
 - affiche_texte, 48
 - affiche_xml_texte, 48
 - duplique_chaine, 48
 - erreur, 49
 - erreur_1s, 49
 - indent, 49
 - indent_step, 49
 - indent_xml, 49
 - nb_ligne, 49
- util.h
 - affiche_balise_fermante, 31
 - affiche_baliseouvrante, 31
 - affiche_element, 31
 - affiche_texte, 31
 - duplique_chaine, 31
 - erreur, 31
 - erreur_1s, 31
- VIRGULE
 - symboles.h, 27
- valeur
 - analysteur_syntaxique.c, 42
- var
 - analysteur_syntaxique.c, 41
 - analysteur_syntaxique.h, 19
 - n_exp_, 9
 - n_instr_, 11
- varDec
 - n_dec_, 7
- varDec_
 - n_dec_, 8
- varExp
 - n_exp_, 8
- variables
 - n_dec_, 8
 - n_prog_, 13
- videlInst
 - n_instr_, 10
- YYTEXT_MAX
 - analysteur_lexical.c, 34
- yyin
 - analysteur_lexical.c, 35
 - compilateur.c, 42
- yylen
 - analysteur_lexical.c, 35
- yylex
 - analysteur_lexical.c, 34
 - analysteur_lexical.h, 15
- yytext
 - analysteur_lexical.c, 35
 - analysteur_syntaxique.c, 42
 - compilateur.c, 42